



# The Sipwise C5 CE Handbook mr8.5.3

Sipwise GmbH

<support@sipwise.com>

# Contents

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Introduction</b>                                 | <b>1</b> |
| 1.1      | About this Handbook . . . . .                       | 1        |
| 1.2      | What is the Sipwise C5 CE? . . . . .                | 1        |
| 1.3      | The Advantages of the Sipwise C5 CE . . . . .       | 1        |
| 1.4      | Who is the Sipwise C5 CE for? . . . . .             | 2        |
| 1.5      | Getting Help . . . . .                              | 2        |
| 1.5.1    | Community Support . . . . .                         | 2        |
| 1.5.2    | Commercial Support . . . . .                        | 2        |
| <b>2</b> | <b>Architecture</b>                                 | <b>3</b> |
| 2.1      | Platforms . . . . .                                 | 3        |
| 2.1.1    | CE Platform . . . . .                               | 3        |
| 2.2      | SIP Signaling and Media Relay . . . . .             | 4        |
| 2.2.1    | SIP Load-Balancer . . . . .                         | 4        |
| 2.2.2    | SIP Proxy/Registrar . . . . .                       | 5        |
| 2.2.3    | SIP Back-to-Back User-Agent (B2BUA) . . . . .       | 5        |
| 2.2.4    | SIP App-Server . . . . .                            | 6        |
| 2.2.5    | Message Routing and Media Relay . . . . .           | 7        |
| 2.3      | Redis Database . . . . .                            | 7        |
| <b>3</b> | <b>Initial Installation</b>                         | <b>8</b> |
| 3.1      | Prerequisites . . . . .                             | 8        |
| 3.2      | Using Sipwise C5 install CD (recommended) . . . . . | 8        |
| 3.3      | Using the Sipwise C5 installer . . . . .            | 9        |
| 3.3.1    | Installing the Operating System . . . . .           | 9        |
| 3.3.2    | Installing the Sipwise C5 . . . . .                 | 10       |
| 3.3.3    | Initial Configuration . . . . .                     | 11       |
| 3.4      | Using a pre-installed virtual machine . . . . .     | 11       |

|          |   |           |
|----------|---|-----------|
| 3.4.1    | Vagrant box for VirtualBox                      | 12        |
| 3.4.2    | VirtualBox image                                | 14        |
| 3.4.3    | VMware image                                    | 15        |
| 3.4.4    | Amazon EC2 image                                | 15        |
| <b>4</b> | <b>Initial System Configuration</b>             | <b>20</b> |
| 4.1      | Network Configuration                           | 20        |
| 4.2      | Apply Configuration Changes                     | 22        |
| 4.3      | Start Securing Your Server                      | 23        |
| 4.4      | Configuring the system-wide editor              | 23        |
| 4.5      | Configuring the Email Server                    | 23        |
| 4.6      | Advanced Network Configuration                  | 24        |
| 4.7      | What's next?                                    | 24        |
| <b>5</b> | <b>VoIP Service Administration Concepts</b>     | <b>25</b> |
| 5.1      | Contacts  | 25        |
| 5.2      | Resellers                                       | 25        |
| 5.3      | SIP Domain                                      | 26        |
| 5.3.1    | Additional SIP Domains                          | 26        |
| 5.4      | Contracts                                       | 27        |
| 5.5      | Customers                                       | 27        |
| 5.5.1    | Residential and SOHO customers                  | 27        |
| 5.5.2    | Business customers with the Cloud PBX service   | 28        |
| 5.5.3    | SIP Trunking                                    | 29        |
| 5.5.4    | Mobile subscribers                              | 29        |
| 5.5.5    | Pre-paid subscribers who use your calling cards | 29        |
| 5.6      | Subscribers                                     | 29        |
| 5.7      | SIP Peerings                                    | 30        |
| <b>6</b> | <b>VoIP Service Configuration Scenario</b>      | <b>32</b> |

|          |  |           |
|----------|--|-----------|
| 6.1      | Creating a SIP Domain                                  | 32        |
| 6.2      | Creating a Customer                                    | 33        |
| 6.3      | Creating a Subscriber                                  | 38        |
| 6.4      | Domain Preferences                                     | 42        |
| 6.5      | Subscriber Preferences                                 | 45        |
| 6.6      | Creating Peerings                                      | 46        |
| 6.6.1    | Creating Peering Groups                                | 46        |
| 6.6.2    | Creating Peering Servers                               | 48        |
| 6.6.3    | Authenticating and Registering against Peering Servers | 60        |
| 6.7      | Configuring Rewrite Rule Sets                          | 62        |
| 6.7.1    | Inbound Rewrite Rules for Caller                       | 65        |
| 6.7.2    | Inbound Rewrite Rules for Callee                       | 67        |
| 6.7.3    | Outbound Rewrite Rules for Caller                      | 68        |
| 6.7.4    | Outbound Rewrite Rules for Callee                      | 69        |
| 6.7.5    | Emergency Number Handling                              | 69        |
| 6.7.6    | Assigning Rewrite Rule Sets to Domains and Subscribers | 71        |
| 6.7.7    | Creating Dialplans for Peering Servers                 | 72        |
| 6.7.8    | Call Routing Verification                              | 72        |
| <b>7</b> | <b>Features</b>  | <b>78</b> |
| 7.1      | About the Admin Web Interface                          | 78        |
| 7.1.1    | Filtering the Lists / Datatables                       | 78        |
| 7.1.2    | Call History   | 79        |
| 7.2      | Managing System Administrators                         | 80        |
| 7.2.1    | Configuring Administrators                             | 80        |
| 7.2.2    | Access Rights of Administrators                        | 82        |
| 7.3      | Access Control for SIP Calls                           | 84        |
| 7.3.1    | Block Lists  | 84        |
| 7.3.2    | NCOS (Network Class of Service) Levels                 | 86        |



---

|       |                                       |     |
|-------|---------------------------------------|-----|
| 7.3.3 | IP Address Restriction                | 94  |
| 7.3.4 | CLI-based Access Control              | 94  |
| 7.3.5 | Call Limit Control                    | 95  |
| 7.4   | Call Forwarding and Call Hunting      | 96  |
| 7.4.1 | Call Forward Types                    | 96  |
| 7.4.2 | Setting a simple Call Forward         | 96  |
| 7.4.3 | Call Forward Destinations             | 97  |
| 7.4.4 | Advanced Call Hunting                 | 98  |
| 7.5   | Call Forking by Q value               | 106 |
| 7.5.1 | The Q value                           | 106 |
| 7.5.2 | The Standard Method                   | 107 |
| 7.5.3 | The Probability Method                | 107 |
| 7.5.4 | Advanced Configurations               | 107 |
| 7.6   | Local Number Porting                  | 108 |
| 7.6.1 | Local LNP Database                    | 108 |
| 7.7   | Emergency Mapping                     | 112 |
| 7.7.1 | Emergency Mapping Description         | 112 |
| 7.7.2 | Emergency Mapping Configuration       | 113 |
| 7.8   | Emergency Priorization                | 119 |
| 7.8.1 | Call-Flow with Emergency Mode Enabled | 120 |
| 7.8.2 | Configuration of Emergency Mode       | 123 |
| 7.8.3 | Activating Emergency Mode             | 124 |
| 7.9   | SIP Message Filtering                 | 125 |
| 7.9.1 | Header Filtering                      | 125 |
| 7.9.2 | Codec Filtering                       | 126 |
| 7.9.3 | Enable History and Diversion Headers  | 127 |
| 7.9.4 | User Agent Filtering                  | 127 |
| 7.10  | SIP Trunking with SIPconnect          | 128 |

|  |     |
|--|-----|
| 7.10.1 User provisioning . . . . .   | 128 |
| 7.10.2 Inbound calls routing . . . . .   | 128 |
| 7.10.3 Number manipulations . . . . .  | 128 |
| 7.10.4 Registration . . . . .  | 131 |
| 7.11 Trusted Subscribers . . . . .   | 132 |
| 7.12 Peer Probing . . . . .  | 132 |
| 7.12.1 Introduction to Peer Probing Feature . . . . .  | 132 |
| 7.12.2 Configuration of Peer Probing . . . . .   | 133 |
| 7.12.3 Monitoring of Peer Probing . . . . .  | 135 |
| 7.12.4 Further Details for Advanced Users . . . . .  | 135 |
| 7.13 Voicemail System . . . . .  | 136 |
| 7.13.1 Accessing the IVR Menu . . . . .  | 136 |
| 7.13.2 IVR Menu Structure . . . . .  | 136 |
| 7.13.3 Type Of Messages . . . . .  | 138 |
| 7.13.4 Folders . . . . .   | 139 |
| 7.13.5 Voicemail Languages Configuration . . . . .   | 139 |
| 7.13.6 Flowcharts with Voice Prompts . . . . .   | 139 |
| 7.14 Configuring Subscriber IVR Language . . . . .   | 144 |
| 7.15 Sound Sets . . . . .  | 144 |
| 7.15.1 Sound_Set and Contract_Sound_Set Usage . . . . .  | 145 |
| 7.15.2 Configuring Early Reject Sound Sets . . . . .   | 145 |
| 7.15.3 Play an announcement on behalf of callee server failure in case of outbound calls . . . . . | 151 |
| 7.16 Conference System . . . . .   | 151 |
| 7.16.1 Configuring Call Forward to Conference . . . . .  | 151 |
| 7.16.2 Configuring Conference Sound Sets . . . . .   | 152 |
| 7.16.3 Joining the Conference . . . . .  | 154 |
| 7.16.4 Conference Flowchart with Voice Prompts . . . . .   | 154 |
| 7.17 Malicious Call Identification (MCID) . . . . .  | 156 |

|  |     |
|--|-----|
| 7.17.1 Setup . . . . .   | 156 |
| 7.17.2 Usage . . . . .   | 157 |
| 7.18 Subscriber Profiles . . . . .                                     | 157 |
| 7.18.1 Subscriber Profile Sets . . . . .                               | 157 |
| 7.19 SIP Loop Detection . . . . .                                      | 160 |
| 7.20 Invoices and Invoice Templates . . . . .                          | 160 |
| 7.20.1 Invoices Management . . . . .                                   | 160 |
| 7.20.2 Invoice Management via REST API . . . . .                       | 162 |
| 7.20.3 Invoice Templates . . . . .                                     | 167 |
| 7.21 Email Reports and Notifications . . . . .                         | 177 |
| 7.21.1 Email events . . . . .  | 177 |
| 7.21.2 Initial template values and template variables . . . . .        | 177 |
| 7.21.3 Subscriber password reset email template . . . . .              | 177 |
| 7.21.4 Administrator password reset email template . . . . .           | 178 |
| 7.21.5 New subscriber notification email template . . . . .            | 178 |
| 7.21.6 Invoice email template . . . . .                                | 179 |
| 7.21.7 Email templates management . . . . .                            | 181 |
| 7.22 The Vertical Service Code Interface . . . . .                     | 183 |
| 7.22.1 Configuration of Vertical Service Codes . . . . .               | 184 |
| 7.22.2 Voice Prompts for Vertical Service Code Configuration . . . . . | 184 |
| 7.23 Handling WebRTC Clients . . . . .                                 | 185 |
| 7.24 XMPP and Instant Messaging . . . . .                              | 186 |
| 7.25 Call Recording . . . . .  | 186 |
| 7.25.1 Introduction to Call Recording Function . . . . .               | 186 |
| 7.25.2 Information on Files and Directories . . . . .                  | 187 |
| 7.25.3 Configuration . . . . .   | 188 |
| 7.25.4 REST API . . . . .  | 192 |
| 7.25.5 Pre-Recording Announcement . . . . .                            | 193 |

|   |            |
|---|------------|
| 7.26 Media Transcoding . . . . .                                | 193        |
| 7.26.1 Overview . . . . .                                       | 193        |
| 7.26.2 Supported Codecs . . . . .                               | 193        |
| 7.26.3 Configuration . . . . .                                  | 194        |
| 7.26.4 T.38 transcoding . . . . .                               | 197        |
| 7.27 Announcement Before Call Setup . . . . .                   | 197        |
| 7.28 Store Recent Calls and Redial . . . . .                    | 198        |
| 7.28.1 Configuring Recent Calls Sound Sets . . . . .            | 198        |
| 7.28.2 Advanced configuration . . . . .                         | 199        |
| 7.29 SMS (Short Message Service) on Sipwise C5 . . . . .        | 199        |
| 7.29.1 Configuration . . . . .                                  | 201        |
| 7.29.2 Monitoring, troubleshooting . . . . .                    | 202        |
| 7.29.3 REST API . . . . .                                       | 209        |
| 7.30 Time sets management . . . . .                             | 210        |
| 7.30.1 Time sets specifications and data description . . . . .  | 210        |
| 7.30.2 Web interface for the time sets . . . . .                | 211        |
| 7.30.3 Web interface for the time set events . . . . .          | 213        |
| 7.30.4 Web interface for time set related to reseller . . . . . | 221        |
| 7.30.5 REST API . . . . .                                       | 222        |
| 7.31 Generation of 181 Call Is Being Forwarded . . . . .        | 223        |
| 7.31.1 Overview . . . . .                                       | 223        |
| 7.31.2 How to enable it . . . . .                               | 224        |
| 7.31.3 How it works . . . . .                                   | 224        |
| <b>8 Customer Self-Care Interface and Menus</b>                 | <b>225</b> |
| 8.1 The Customer Self-Care Web Interface . . . . .              | 225        |
| 8.1.1 Login Procedure . . . . .                                 | 225        |
| 8.1.2 Site Customization . . . . .                              | 225        |
| 8.2 The Voicemail Menu . . . . .                                | 231        |

|                                      |            |
|--------------------------------------|------------|
| <b>9 Billing Configuration</b>       | <b>232</b> |
| 9.1 Billing Profiles                 | 232        |
| 9.1.1 Creating Billing Profiles      | 232        |
| 9.1.2 Creating Billing Fees          | 234        |
| 9.1.3 Creating Off-Peak Times        | 237        |
| 9.2 Fraud Detection and Locking      | 239        |
| 9.2.1 Fraud Lock Levels              | 240        |
| 9.3 Notes on Billing and Call Rating | 240        |
| 9.4 Billing Data Export              | 241        |
| 9.4.1 Glossary of Terms              | 241        |
| 9.4.2 File Name Format               | 242        |
| 9.4.3 File Format                    | 242        |
| 9.4.4 File Transfer                  | 255        |
| <b>10 Corosync/Pacemaker</b>         | <b>256</b> |
| 10.1 Overview                        | 256        |
| 10.2 Migration                       | 256        |
| 10.2.1 Rollback                      | 257        |
| 10.3 Corosync                        | 257        |
| 10.3.1 Quorum                        | 257        |
| 10.4 Pacemaker                       | 257        |
| 10.5 Query Status                    | 258        |
| 10.6 Config Management               | 259        |
| 10.6.1 General Concepts              | 261        |
| 10.6.2 Resources                     | 261        |
| 10.6.3 Cluster Options               | 264        |
| 10.6.4 Failure Counts                | 265        |
| 10.6.5 Resource Scores               | 266        |
| 10.7 Common Tasks                    | 267        |

---

|   |            |
|---|------------|
| 10.7.1 Takeover and Standby . . . . .                                     | 267        |
| 10.7.2 Node Status (Online/Standby) . . . . .                             | 267        |
| 10.7.3 Maintenance Mode . . . . .   | 268        |
| 10.7.4 CLI Alternatives . . . . .   | 268        |
| <b>11 Provisioning REST API Interface</b>                                 | <b>269</b> |
| 11.1 API Workflows for Customer and Subscriber Management . . . . .       | 269        |
| 11.2 API performance considerations . . . . .                             | 274        |
| <b>12 Configuration Framework</b>   | <b>275</b> |
| 12.1 Configuration templates . . . . .                                    | 275        |
| 12.1.1 .tt2, .customtt.tt2 and .patchtt.tt2 files . . . . .               | 275        |
| 12.1.2 Using patchtt for generation of a relevant customtt file . . . . . | 278        |
| 12.1.3 .prebuild and .postbuild files . . . . .                           | 279        |
| 12.1.4 .services files . . . . .  | 280        |
| 12.2 config.yml, constants.yml and network.yml files . . . . .            | 281        |
| 12.3 ngcpcfg and its command line options . . . . .                       | 281        |
| 12.3.1 apply . . . . .  | 281        |
| 12.3.2 build . . . . .  | 281        |
| 12.3.3 commit . . . . .   | 282        |
| 12.3.4 decrypt . . . . .  | 282        |
| 12.3.5 diff . . . . .   | 282        |
| 12.3.6 encrypt . . . . .  | 282        |
| 12.3.7 help . . . . .   | 282        |
| 12.3.8 initialise . . . . .   | 282        |
| 12.3.9 services . . . . .   | 282        |
| 12.3.10status . . . . .   | 283        |
| <b>13 Network Configuration</b>   | <b>284</b> |
| 13.1 General Structure . . . . .  | 284        |

|  |            |
|--|------------|
| 13.1.1 Available Host Options . . . . .                            | 284        |
| 13.1.2 Interface Parameters . . . . .                              | 285        |
| 13.2 Advanced Network Configuration . . . . .                      | 286        |
| 13.2.1 Additional entries in <code>/etc/hosts</code> . . . . .     | 287        |
| 13.2.2 Extra SIP Sockets . . . . .                                 | 288        |
| 13.2.3 Extra SIP and RTP Sockets . . . . .                         | 289        |
| 13.2.4 Alternative RTP Interface Selection Using ICE . . . . .     | 290        |
| 13.2.5 Extended RTP Port Range Using Multiple Interfaces . . . . . | 291        |
| <b>14 Licenses</b>   | <b>292</b> |
| <b>15 Software Upgrade</b>   | <b>293</b> |
| 15.1 Release Notes . . . . .                                       | 293        |
| 15.2 Overview . . . . .  | 293        |
| 15.3 Pre-upgrade checks . . . . .                                  | 293        |
| 15.3.1 Log into the C5 server . . . . .                            | 294        |
| 15.3.2 Check the overall system status . . . . .                   | 294        |
| 15.3.3 Evaluate and update custom modifications . . . . .          | 294        |
| 15.3.4 Check system integrity . . . . .                            | 295        |
| 15.3.5 Check the configuration framework status . . . . .          | 295        |
| 15.4 Pre-upgrade steps . . . . .                                   | 296        |
| 15.4.1 <code>ngcp-upgrade</code> options . . . . .                 | 296        |
| 15.4.2 Preparing for maintenance mode . . . . .                    | 297        |
| 15.4.3 Set the proper software repositories . . . . .              | 297        |
| 15.5 Upgrading Sipwise C5 CE . . . . .                             | 297        |
| 15.6 Post-upgrade steps . . . . .                                  | 298        |
| 15.6.1 Disabling maintenance mode . . . . .                        | 298        |
| 15.6.2 Post-upgrade checks . . . . .                               | 298        |
| 15.7 Applying the Latest Hotfixes . . . . .                        | 298        |
| 15.7.1 Apply hotfixes . . . . .                                    | 298        |

|  |            |
|--|------------|
| 15.7.2 Recheck or update the custom configuration templates . . . . .                    | 299        |
| <b>16 Backup, Recovery and Database Maintenance</b>                                      | <b>300</b> |
| 16.1 Sipwise C5 Backup . . . . .   | 300        |
| 16.1.1 What data to back up . . . . .  | 300        |
| 16.2 Recovery . . . . .  | 300        |
| 16.3 Reset Database . . . . .  | 301        |
| 16.4 Accounting Data (CDR) Cleanup . . . . .   | 301        |
| 16.4.1 Cleanuptools Configuration . . . . .  | 301        |
| 16.4.2 Accounting Database Cleanup . . . . .   | 301        |
| 16.4.3 Exported CDR Cleanup . . . . .  | 304        |
| <b>17 Platform Security, Performance and Troubleshooting</b>                             | <b>306</b> |
| 17.1 Sipwise SSH access to Sipwise C5 . . . . .  | 306        |
| 17.2 Firewalling . . . . .   | 307        |
| 17.2.1 Firewall framework . . . . .  | 307        |
| 17.2.2 Sipwise C5 firewall configuration . . . . .                                       | 308        |
| 17.2.3 IPv4 System rules . . . . .   | 309        |
| 17.2.4 Custom rules . . . . .  | 312        |
| 17.2.5 Example firewall configuration section . . . . .                                  | 313        |
| 17.3 Password management . . . . .   | 313        |
| 17.3.1 The "root" account . . . . .  | 313        |
| 17.3.2 The "administrator" account . . . . .   | 314        |
| 17.3.3 The "cdrexport" account . . . . .   | 314        |
| 17.3.4 The MySQL "root" user . . . . .   | 314        |
| 17.3.5 The "ngcpssoap" account . . . . .   | 314        |
| 17.4 Remote <i>root</i> logins via SSH . . . . .   | 314        |
| 17.5 <i>sudo-io</i> : logging input/output of commands run through <i>sudo</i> . . . . . | 315        |
| 17.6 SSL certificates . . . . .  | 316        |
| 17.7 Securing your Sipwise C5 against SIP attacks . . . . .                              | 317        |



|   |            |
|---|------------|
| 17.7.1 Denial of Service . . . . .                                | 317        |
| 17.7.2 Bruteforcing SIP credentials . . . . .                     | 318        |
| 17.8 Topology Hiding . . . . .                                    | 319        |
| 17.8.1 Introduction to Topology Hiding on NGCP . . . . .          | 319        |
| 17.8.2 Topology Masking Mechanism . . . . .                       | 319        |
| 17.8.3 Topology Hiding Mechanism . . . . .                        | 320        |
| 17.9 System Requirements and Performance . . . . .                | 321        |
| 17.10 Troubleshooting . . . . .                                   | 324        |
| 17.10.1 Collecting call information from logs . . . . .           | 326        |
| 17.10.2 Collecting SIP traces . . . . .                           | 326        |
| <b>18 Monitoring and Alerting</b>                                 | <b>328</b> |
| 18.1 Internal Monitoring . . . . .                                | 328        |
| 18.1.1 System monitoring via Telegraf . . . . .                   | 328        |
| 18.1.2 Sipwise C5 specific monitoring via ngcp-witnessd . . . . . | 328        |
| 18.1.3 Monitoring data in InfluxDB . . . . .                      | 328        |
| 18.2 Statistics Dashboard . . . . .                               | 329        |
| <b>A Basic Call Flows</b>   | <b>330</b> |
| A.1 General Call Setup . . . . .                                  | 330        |
| A.2 Endpoint Registration . . . . .                               | 331        |
| A.3 Basic Call . . . . .  | 334        |
| A.4 Session Keep-Alive . . . . .                                  | 335        |
| A.5 Voicebox Calls . . . . .                                      | 336        |
| <b>B Sipwise C5 configs overview</b>                              | <b>338</b> |
| B.1 config.yml Overview . . . . .                                 | 338        |
| B.1.1 apps . . . . .  | 338        |
| B.1.2 asterisk . . . . .  | 338        |
| B.1.3 autoprovider . . . . .                                      | 339        |

|        |   |     |
|--------|---|-----|
| B.1.4  | backuptools   | 340 |
| B.1.5  | cdrexport   | 341 |
| B.1.6  | cleanuptools  | 342 |
| B.1.7  | cluster_sets  | 343 |
| B.1.8  | database  | 343 |
| B.1.9  | faxserver   | 344 |
| B.1.10 | general   | 344 |
| B.1.11 | heartbeat   | 345 |
| B.1.12 | intercept   | 345 |
| B.1.13 | kamailio  | 345 |
| B.1.14 | ngcp-mediator   | 359 |
| B.1.15 | modules   | 360 |
| B.1.16 | monitoring  | 360 |
| B.1.17 | nginx   | 361 |
| B.1.18 | ntp   | 361 |
| B.1.19 | ossbss  | 362 |
| B.1.20 | pbx (only with additional cloud PBX module installed) | 363 |
| B.1.21 | prosody   | 364 |
| B.1.22 | pushd   | 364 |
| B.1.23 | qos   | 367 |
| B.1.24 | ngcp-rate-o-mat                                       | 367 |
| B.1.25 | redis   | 367 |
| B.1.26 | reminder  | 368 |
| B.1.27 | rsyslog   | 368 |
| B.1.28 | rtpproxy  | 369 |
| B.1.29 | security  | 371 |
| B.1.30 | sems  | 372 |
| B.1.31 | sms   | 374 |

|  |            |
|--|------------|
| B.1.32 sshd . . . . .                  | 375        |
| B.1.33 sudo . . . . .                  | 375        |
| B.1.34 telegraf . . . . .              | 376        |
| B.1.35 ngcp-witnessd . . . . .         | 376        |
| B.1.36 www_admin . . . . .             | 378        |
| B.2 constants.yml Overview . . . . .   | 380        |
| B.3 network.yml Overview . . . . .     | 380        |
| <b>C MariaDB encryption</b>            | <b>382</b> |
| C.1 Overview . . . . .                 | 382        |
| C.2 Configuration . . . . .            | 382        |
| C.3 What is not encrypted . . . . .    | 382        |
| C.4 Data restoration remarks . . . . . | 383        |
| <b>D NGCP Internals</b>                | <b>384</b> |
| D.1 Pending reboot marker . . . . .    | 384        |
| D.2 Redis id constants . . . . .       | 384        |
| D.3 InfluxDB monitoring keys . . . . . | 385        |
| D.4 Preferences . . . . .              | 386        |
| D.4.1 Tables . . . . .                 | 386        |
| D.4.2 Columns . . . . .                | 386        |
| D.4.3 Enum . . . . .                   | 387        |
| <b>E Kamailio pv_headers module</b>    | <b>389</b> |
| E.1 Module overview . . . . .          | 389        |
| E.2 Template changes . . . . .         | 389        |
| E.3 Module documentation . . . . .     | 390        |
| E.3.1 Parameters . . . . .             | 390        |
| E.3.2 Functions . . . . .              | 391        |
| E.3.3 Pseudovariables . . . . .        | 394        |

---

|          |   |            |
|----------|---|------------|
| <b>F</b> | <b>Extra Configuration Scenarios</b>        | <b>399</b> |
| F.1      | AudioCodes devices workaround . . . . .     | 399        |
| F.2      | "Debug Proxy" for troubleshooting . . . . . | 399        |

# 1 Introduction

## 1.1 About this Handbook

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

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

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

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

## 1.2 What is the Sipwise C5 CE?

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

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

The Sipwise C5 CE can be installed in a few steps within a couple of minutes and requires no knowledge about configuration files of specific software components.

## 1.3 The Advantages of the Sipwise C5 CE

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

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

After downloading and starting the installer, it will fetch and install all the required Debian packages from the relevant Debian repositories. The installed applications are managed by the Sipwise C5 Configuration Framework. This configuration framework makes it possible to change low-level system parameters in a single place, so Sipwise C5 administrators don't need to have any knowledge of dozens of different configuration files from different packages. This provides a very easy and bullet-proof way of

operating, changing and tweaking an otherwise quite complex system.

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

## 1.4 Who is the Sipwise C5 CE for?

The Sipwise C5 CE is specifically tailored to companies and engineers trying to start or experiment with a fully-featured SIP-based VoIP service without having to go through the steep learning curve of SIP signalling. It integrates the different building blocks to make them work together in a reasonable way and implements the missing components to build a business on top of that.

In the past, creating a business-ready VoIP service included installation and configuration of SIP software like Asterisk, OpenSER, Kamailio, etc., which can get quite difficult when it comes to implementing advanced features. It required implementing different web interfaces, billing engines and connectors to existing OSS/BSS infrastructure. These things are now obsolete due to the Sipwise C5 CE, which covers all these requirements.

## 1.5 Getting Help

### 1.5.1 Community Support

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

### 1.5.2 Commercial Support

If you need professional help setting up and maintaining the Sipwise C5 CE, send an email to [sales@sipwise.com](mailto:sales@sipwise.com).

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

## 2 Architecture

### 2.1 Platforms

#### 2.1.1 CE Platform

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

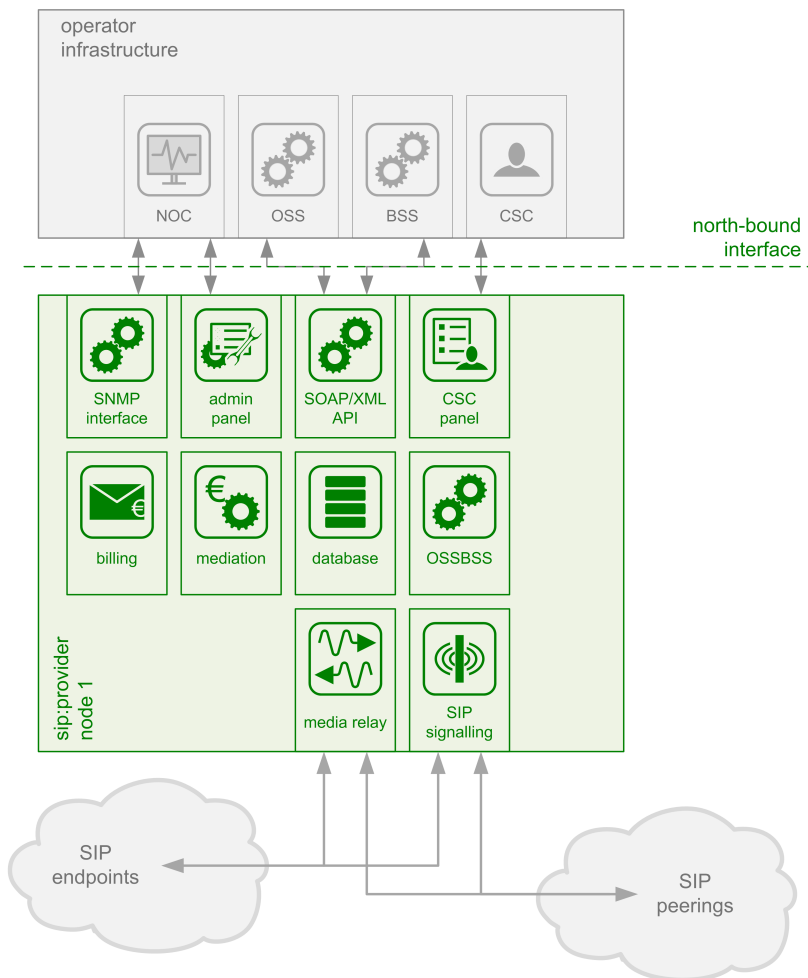


Figure 1: CE Architecture Overview

The main building blocks of Sipwise C5 are:

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

## 2.2 SIP Signaling and Media Relay

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

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

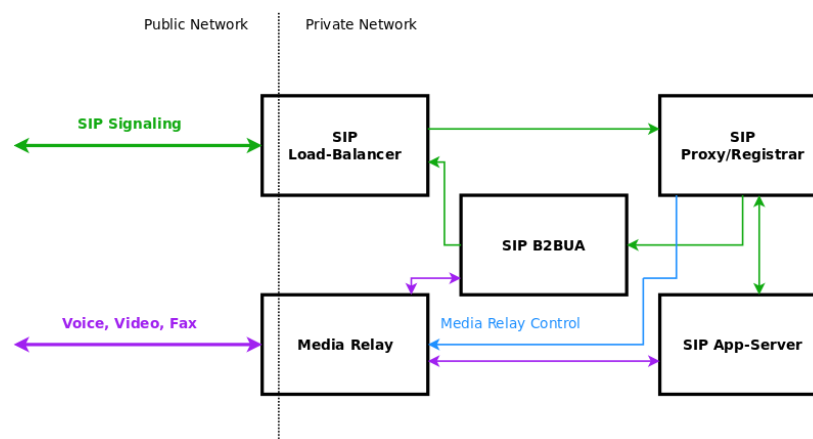


Figure 2: SIP and Media Relay Components

### 2.2.1 SIP Load-Balancer

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

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

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

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

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



---

**Tip**

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

---

### 2.2.2 SIP Proxy/Registrar

The SIP proxy/registrar (or short *proxy*) is the work-horse of Sipwise C5. It'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 Sipwise C5, which is especially important to traverse NAT.

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

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

---

**Tip**

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

---

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

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

The software part used for this element is SEMS.

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

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

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

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

---

**Tip**

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

---

## 2.2.4 SIP App-Server

The SIP App-Server is an Asterisk instance used for voice applications like Voicemail and Reminder Calls. Asterisk uses the MySQL database as a message spool for voicemail, so it 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 Sipwise C5 internal UUIDs for each subscriber. That way a SIP subscriber can have multiple E164 phone numbers, but all of them terminate in the same voicebox.

The App-Server listens on the internal interface on UDP port 5070 for SIP messages and by default uses media ports in the range from UDP port 10000 to 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 `ngcp-service start asterisk`, `ngcp-service stop asterisk` and `ngcp-service restart asterisk`. Its status can be queried by executing `ngcp-service status asterisk` or `ngcp-service summary | grep "asterisk"`.

---

### 2.2.5 Message Routing and Media Relay

The Media Relay (also called *rtengine*) is a Kernel-based packet relay, which is controlled by the SIP proxy. For each media stream (e.g. a voice and/or video stream), it maintains a pair of ports in the range of port number 30000 to 40000. When the media streams are negotiated, *rtengine* opens the ports in user-space and starts relaying the packets to the addresses announced by the endpoints. If packets arrive from different source addresses than announced in the SDP body of the SIP message (e.g. in case of NAT), the source address is implicitly changed to the address the packets are received from. Once the call is established and the *rtengine* 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 *rtengine* 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 *rtengine* 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 *rtengine* 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 *rtengine* configuration file is `/etc/ngcp-config/templates/etc/default/ngcp-rtengine-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 `ngcp-service start rtengine`, `ngcp-service stop rtengine` and `ngcp-service restart rtengine`. Its status can be queried by executing `ngcp-service status rtengine` or `ngcp-service summary | grep "rtengine"`.

---

## 2.3 Redis Database

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

## 3 Initial Installation

### 3.1 Prerequisites

For an initial installation of Sipwise C5 , it is mandatory that your production environment meets the following criteria:

#### HARDWARE REQUIREMENTS

- Recommended: Dual-core, x86\_64 compatible, 3GHz, 4GB RAM, 128GB HDD
- Minimum: Single-core, x86\_64 compatible, 1GHz, 2GB RAM, 24GB HDD

#### SUPPORTED OPERATING SYSTEMS

- Debian 10 (buster) 64-bit

#### INTERNET CONNECTION

- Hardware needs connection to the Internet



#### Important

Only **Debian 10 (buster) 64-bit** is currently supported as a host system for Sipwise C5 .

---



#### Important

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

---

### 3.2 Using Sipwise C5 install CD (recommended)

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



#### Important

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

---

You can install the current Sipwise C5 CE version mr8.5.3 using [install CD image](#) (checksums: [sha1](#) , [md5](#)).

---

**Important**

The Sipwise C5 install CD automatically takes care of partitioning, any present data will be overwritten! While the installer prompts for the disk that should be used for installation before its actual execution, it's strongly recommended to boot the ISO in an environment with empty disks or disks that you don't plan to use for anything else than the newly installed Sipwise C5 system.

---

**Tip**

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

---

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

### 3.3 Using the Sipwise C5 installer

#### 3.3.1 Installing the Operating System

You need to install Debian 10 (buster) 64-bit on the server. A **basic** installation without any additional task selection (like *Desktop System*, *Web Server* etc.) is sufficient.

---

**Tip**

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

---

**Important**

If you use other kinds of installation media (e.g. provided by your hosting provider), prepare for some issues that might come up during installation. For example, you might be forced to manually resolve package dependencies in order to install Sipwise C5 . Therefore, it is **HIGHLY RECOMMENDED** to use a clean Debian installation to simplify the installation process.

---

**Note**

If you installed your system using the Debian CDs/DVDs (so neither using Sipwise C5 install CD nor **the Debian Netinstall ISO**) `apt-get` might prompt to insert disk to proceed during Sipwise C5 installation. The prompt won't be visible for you and installation hangs. Please disable the cdrom entries in `/etc/apt/sources.list` and enable a Debian mirror (e.g. <https://deb.debian.org/debian/>) instead.

---

#### 3.3.1.1 Using special Debian setups

If you plan to install Sipwise C5 on Virtual Hosting Providers like *Dreamhost* with their provided Debian installer, you might need to manually prepare the system for Sipwise C5 installation, otherwise the installer will fail installing certain package versions required

---

to function properly.

### Using Dreamhost Virtual Private Server

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

---

#### Note

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

---

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



#### Warning

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

---

### 3.3.2 Installing the Sipwise C5

Download and install the latest *Sipwise C5* installer package:

```
PKG=ngcp-installer-mr8.5.3.deb
wget http://deb.sipwise.com/spce/${PKG}
dpkg -i ${PKG}
```

Run the installer as root user:

```
ngcp-installer
```

---

#### Note

You can find the previous versions of *Sipwise C5* installer package [here](#).

---

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

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

```
Installation finished. Thanks for choosing Sipwise C5 Community Edition.  
Please reboot the server to continue with the configuration.
```

**Warning**

Be aware that all services will be disabled. If you need a specific service - re-enable it when the initial configuration is done.

---

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

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

### 3.3.3 Initial Configuration

After the installation has finished successfully and the server gets rebooted it is necessary to perform the initial configuration:

**Warning**

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

---

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

The tool will ask you to confirm the network configuration which is based on the current one. Read **carefully** the information printed on screen, and if you agree, proceed by typing `y`. If you want to change these parameters, you can edit the file `/etc/ngcp-installer/config_deploy.inc` and adjust the variables with the desired values.

If everything goes well, you should see the message:

```
System was successfully configured, now you have the best VoIP software.
```

## 3.4 Using a pre-installed virtual machine

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

### 3.4.1 Vagrant box for VirtualBox

**Vagrant** is an open-source software for creating and configuring virtual development environments. Sipwise provides a so called Vagrant base box for your service, to easily get direct access to your own Sipwise C5 Virtual Machine without any hassles.

---

#### Note

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

---

Get your copy of Sipwise C5 by running:

```
vagrant init spce-mr8.5.3 https://deb.sipwise.com/spce/images/mr8.5.3/sip_provider_CE_mr8 ↔
    .5.3_vagrant.box
vagrant up
```

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

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

Use the following command to access the terminal:

```
vagrant ssh
```

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

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

---

#### Note

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

---

Table 1: Vagrant based VirtualBox VM interfaces:

| Description             | Host-only address   | LAN address   | Notes                            |
|-------------------------|---|---|----------------------------------|
| SSH                     | ssh://127.0.0.1:2222  | ssh://a.b.c.d:22 or<br>ssh://x.y.z.p:2222   | Also available via "vagrant ssh" |
| Administrator interface | <a href="https://127.0.0.1:1443/login/admin">https://127.0.0.1:1443/login/admin</a> | <a href="https://a.b.c.d:1443/login/admin">https://a.b.c.d:1443/login/admin</a> or<br><a href="https://x.y.z.p:1443/login/admin">https://x.y.z.p:1443/login/admin</a> |                                  |



Table 1: (continued)

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

**Note**

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

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

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

To turn off your Sipwise C5 virtual machine, just type:

```
vagrant halt
```

To completely remove Sipwise C5 virtual machine, use:

```
vagrant destroy
vagrant box remove spce-mr8.5.3
```

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

Vagrant usage tips:

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

```
vagrant ssh-config
```

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

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

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

### 3.4.2 VirtualBox image

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

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

On Linux, you can do it like this:

```
tar xvf sip_provider_CE_mr8.5.3_virtualbox.ova
```

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

Considerations when using this virtual machine:

- You will need a 64bit guest capable VirtualBox setup.

- The root password is *sipwise*
- You should use *bridge mode* networking (adjust your bridging interface in the virtual machine configuration) to avoid having Sipwise C5 behind NAT.
- You'll need to adjust your timezone and keyboard layout.
- The network configuration is set to DHCP. You'll need to change it to the appropriate static configuration.
- As the virtual image is a static file, it won't contain the most updated versions of our software. Please upgrade the system via apt as soon as you boot it for the first time.

### 3.4.3 VMware image

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

Considerations when using this virtual machine:

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

### 3.4.4 Amazon EC2 image

Sipwise provides AMI (Amazon Machine Images) images in all Amazon EC2 regions for the latest and LTS Sipwise C5 releases. Please find the appropriate AMI ID for your region in [release announcement](#).

---

#### Note

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

---

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

Choose "Launch Instance":

## Create Instance

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

[Launch Instance](#)

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

Figure 3: Launch Amazon EC2 Instance for your region

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

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

### Step 1: Choose an Amazon Machine Image (AMI) [Cancel and Exit](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

|                       |   |                          |
|-----------------------|---|--------------------------|
| Quick Start           | Q ami-37b87340  | < < 1 to 1 of 1 AMIs > > |
| My AMIs               |   |                          |
| AWS Marketplace       |  <b>ngcp-ce-mr3.3.1</b> - ami-37b87340         | <a href="#">Select</a>   |
| <b>Community AMIs</b> | Official sip:provider CE AMI for release mr3.3.1.3 [2014-06-23_20:51]<br>Root device type: ebs Virtualization type: paravirtual | 64-bit                   |

Figure 4: Choose an image (different for each region)

Select the Instance Type you want to use for running Sipwise C5 (recommended:  $\geq 4$ GB RAM):

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

## Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation [Show/Hide Columns](#)

Currently selected: t2.medium (Variable ECUs, 2 vCPUs, 2.3 GHz, Intel Broadwell E5-2686v4, 4 GiB memory, EBS only)

|                                     | Family          | Type                           | vCPUs | Memory (GiB) | Instance Storage (GB) | EBS-Optimized Available | Network Performance | IPv6 Support |
|-------------------------------------|-----------------|--------------------------------|-------|--------------|-----------------------|-------------------------|---------------------|--------------|
| <input type="checkbox"/>            | General purpose | t2.nano                        | 1     | 0.5          | EBS only              | -                       | Low to Moderate     | Yes          |
| <input type="checkbox"/>            | General purpose | t2.micro<br>Free tier eligible | 1     | 1            | EBS only              | -                       | Low to Moderate     | Yes          |
| <input type="checkbox"/>            | General purpose | t2.small                       | 1     | 2            | EBS only              | -                       | Low to Moderate     | Yes          |
| <input checked="" type="checkbox"/> | General purpose | t2.medium                      | 2     | 4            | EBS only              | -                       | Low to Moderate     | Yes          |
| <input type="checkbox"/>            | General purpose | t2.large                       | 2     | 8            | EBS only              | -                       | Low to Moderate     | Yes          |
| <input type="checkbox"/>            | General purpose | t2.xlarge                      | 4     | 16           | EBS only              | -                       | Moderate            | Yes          |
| <input type="checkbox"/>            | General purpose | t2.2xlarge                     | 8     | 32           | EBS only              | -                       | Moderate            | Yes          |

Figure 5: Choose Amazon EC2 instance

### Tip

Do not forget to tune necessary Sipwise C5 performance parameters depending on Amazon EC2 instance type and performance you are looking for. Find more information about Sipwise C5 performance tuning in Section 17.9.

Run through next configuration options

- Configure Instance: optional (no special configuration required from Sipwise C5)
- Add Storage: choose  $\geq 8$ GB disk size (no further special configuration required from Sipwise C5)
- Tag Instance: optional (no special configuration required from Sipwise C5)
- Configure Security Group: create a new security group, using the following rules:
  - TCP port 22 (SSH)
  - TCP port 443 (HTTPS/CSC)
  - TCP port 1443 (Admin Panel)
  - TCP port 5060 (SIP/TCP)
  - UDP port 5060 (SIP/UDP)
  - TCP port 5061 (SIP/TLS)

- TCP port 5222 (SIP)
- TCP port 5269 (SIP)
- UDP port 30000:44999 (RTP)

**Note**

Please feel free to restrict the *Source* options in your Security Group to your own (range of) IP addresses, especially for SSH and Admin Panel!

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

**Step 6: Configure Security Group**

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

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

Security group name:

Description:

| Type ⓘ       | Protocol ⓘ | Port Range ⓘ | Source ⓘ                 | Description ⓘ |   |
|--------------|------------|--------------|--------------------------|---------------|---|
| SSH ▾        | TCP        | 22           | Custom ▾ 0.0.0.0/0, ::/0 | SSH           | ✕ |
| HTTPS ▾      | TCP        | 443          | Custom ▾ 0.0.0.0/0, ::/0 | HTTPS/CSC     | ✕ |
| Custom TCf ▾ | TCP        | 1443         | Custom ▾ 0.0.0.0/0, ::/0 | Admin Panel   | ✕ |
| Custom TCf ▾ | TCP        | 5060         | Custom ▾ 0.0.0.0/0, ::/0 | SIP/TCP       | ✕ |
| Custom UDI ▾ | UDP        | 5060         | Custom ▾ 0.0.0.0/0, ::/0 | SIP/UDP       | ✕ |
| Custom TCf ▾ | TCP        | 5061         | Custom ▾ 0.0.0.0/0, ::/0 | SIP/TLS       | ✕ |
| Custom TCf ▾ | TCP        | 5222         | Custom ▾ 0.0.0.0/0, ::/0 | SIP           | ✕ |
| Custom TCf ▾ | TCP        | 5269         | Custom ▾ 0.0.0.0/0, ::/0 | SIP           | ✕ |
| Custom UDI ▾ | UDP        | 30000-44999  | Custom ▾ 0.0.0.0/0, ::/0 | RTP           | ✕ |

Figure 6: Configure Security Group

Finally review instance in the last step (Review Instance Launch) and press Launch button.

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

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

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

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

Create a new key pair

Key pair name

sip-provider-ce

Download Key Pair



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

Cancel

Launch Instances

Figure 7: Choose a key pair to access the system

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

The screenshot shows the AWS Management Console interface for the EC2 dashboard. On the left, there is a navigation menu with options like 'EC2 Dashboard', 'Events', 'Tags', 'Reports', 'Limits', 'INSTANCES', 'Instances', 'Spot Requests', and 'Reserved Instances'. The main area displays a table of instances. At the top, there are buttons for 'Launch Instance', 'Connect', and 'Actions'. Below that, there is a filter section with 'All instances' and 'All instance types' dropdowns, and a search box containing 'i-ab68c7e8'. The instance table has columns for Name, Instance ID, Instance Type, Availability, Instance State, Status Checks, Alarm Sta, Public DNS, and Public IP. One instance is listed with ID 'i-ab68c7e8', type 'm3.medium', availability 'eu-west-1b', state 'running', and public IP '54.195.40.1'.

| Name | Instance ID | Instance Type | Availability | Instance State | Status Checks | Alarm Sta | Public DNS              | Public IP   |
|------|-------------|---------------|--------------|----------------|---------------|-----------|-------------------------|-------------|
|      | i-ab68c7e8  | m3.medium     | eu-west-1b   | running        | Initializing  | None      | ec2-54-195-40-219.eu... | 54.195.40.1 |

Figure 8: Running Amazon EC2 Sipwise C5 instance

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

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

DNS name/IP address the system got assigned.

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

Now you can increase your privileges to user *root* for further system configuration:

```
sudo -s
```

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

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

and apply your changes:

```
ngcpcfg apply 'add advertised-ip on interface eth0'
```

Now feel free to use your newly started Amazon EC2 Sipwise C5 instance!



### Warning

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

---

## 4 Initial System Configuration

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

### 4.1 Network Configuration

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

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



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

If you want to enable IPv6 as well, you have to set the address on the proper interface as well, like this (assuming you have an IPv6 address `fd5a:5cc1:23:4:0:0:0:1f` on interface `eth0`):

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

---

### Tip

Always use a full IPv6 address with 8 octets. Leaving out zero octets (e.g. `FD5A:5CC1:23:4::1F`) is **not allowed**.

---



### Important

You should use the IPv6 address in **upper-case** because LB (kamailio) handles the IPv6 addresses internally in upper-case format.

---

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

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

The following configuration shows C5 running in the internal 192.168.0.0/24 network behind the NAT:

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

```
- eth0
  lo:
...

```

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

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

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

```
# File autogenerated by ngcpcfg

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

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

```

Reboot the server to apply the new network configuration:

```
reboot
```

## 4.2 Apply Configuration Changes

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

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

---

### Tip

At this point, your system is ready to serve.

---

### 4.3 Start Securing Your Server

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

```
passwd cdrexport
```

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

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

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

### 4.4 Configuring the system-wide editor

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

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

---

#### Tip

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

---

### 4.5 Configuring the Email Server

The Sipwise C5 installer will install *mailx* (which has *Exim4* as MTA as a default dependency) on the system, however the MTA is not configured by the installer. If you want to use the *Voicemail-to-Email* feature of the Voicebox, you need to configure your MTA properly. If you are fine to use the default MTA *Exim4*, execute the following command:

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

**Important**

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

---

## 4.6 Advanced Network Configuration

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

## 4.7 What's next?

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

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

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

## 5 VoIP Service Administration Concepts

### 5.1 Contacts

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

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

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

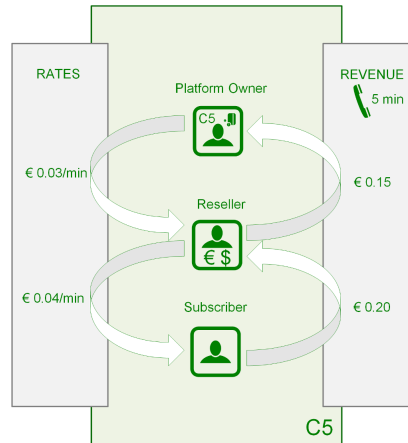
Note that the only required contact field is **email**. For contacts associated with customers, it will be used for sending invoices and notifications such as password reset, new subscriber creation and others. A contact for a subscriber is created automatically but only if you specify an email address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

### 5.2 Resellers

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

Let us consider an example:

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



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

### 5.3 SIP Domain

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

#### 5.3.1 Additional SIP Domains

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

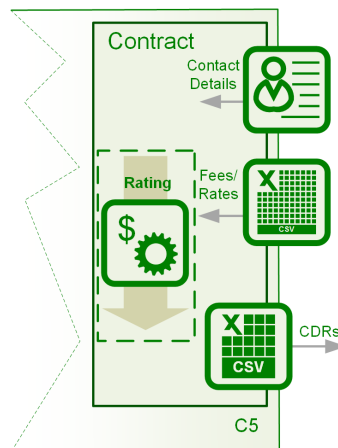
- A dedicated SIP domain is *suggested* per CloudPBX customer.
- A separate SIP domain may be dedicated to every whitelabel reseller.
- Multiple SIP domains may be used to provide services in different countries or regions.
- Multiple SIP domains may be used to brand your own services.

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

## 5.4 Contracts

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

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



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

## 5.5 Customers

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

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

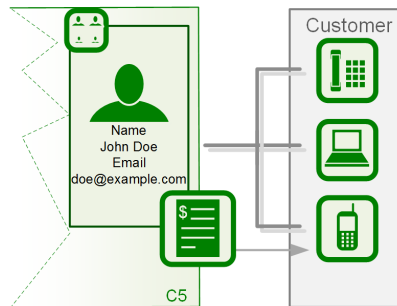
Here are two common examples of the customer model:

### 5.5.1 Residential and SOHO customers

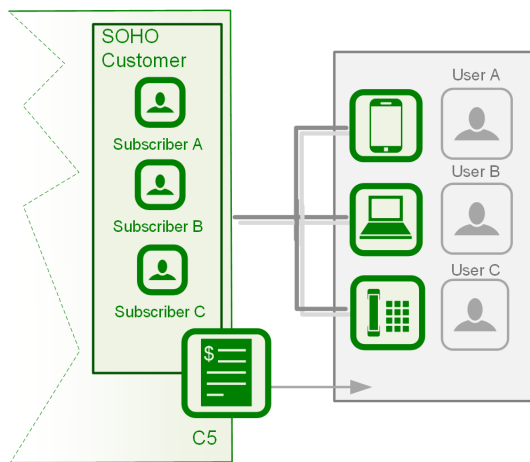
With this service you provide your residential and SOHO customers with one or multiple numbers and offer the service on a post-paid basis.

For a residential customer you usually create one *customer* entity with one *subscriber* under it. A residential customer can register multiple devices with the same number thus having a convenient Viber or Skype-like service: any device can be used to make a

call and all of them will ring simultaneously when there is an incoming call. At the end of the billing period, you send an invoice to the customer.

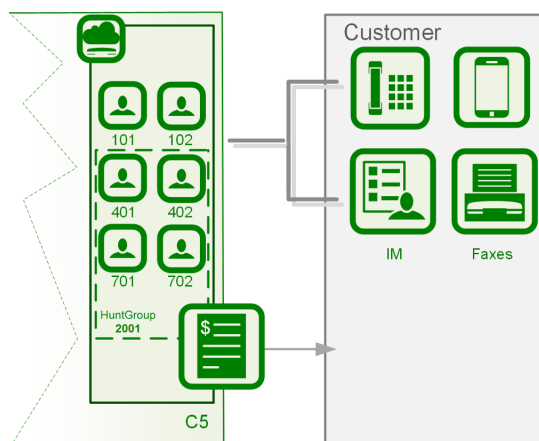


For SOHO customers you usually create multiple subscribers under the same customer and assign every subscriber a dedicated number to allow users make and receive calls. A common invoice will contain calls of all the subscribers.



### 5.5.2 Business customers with the Cloud PBX service

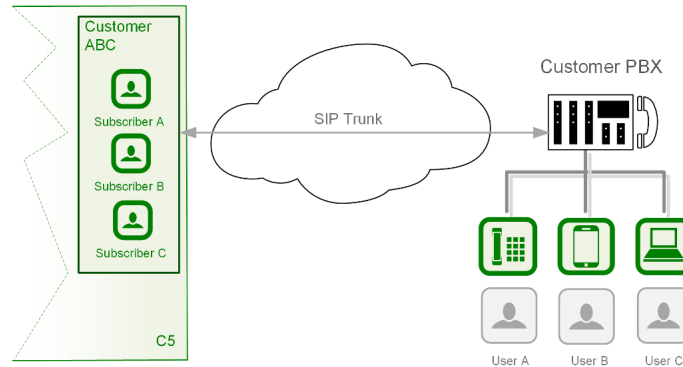
In this case you create a Customer and all the required entities under it to reflect the company's structure: subscribers, extensions, hunt groups, auto-attendant menus, etc.





### 5.5.3 SIP Trunking

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



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

### 5.5.4 Mobile subscribers

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

### 5.5.5 Pre-paid subscribers who use your calling cards

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

## 5.6 Subscribers

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

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

| Service               | Subscriber Type    | Purpose  | Features   |
|-----------------------|--------------------|--|--|
| Residential           | Regular subscriber | A regular VoIP service   | Requires a DID number to receive calls from outside of your network        |
| Enterprise (CloudPBX) | Pilot subscriber   | A base number for the enterprise customer; Lists all extra numbers (aliases) | Configures the rest of customer subscribers in its self-care web interface |

| Service   | Subscriber Type       | Purpose   | Features  |
|-----------|-----------------------|---|---|
|           | Extension             | Extra numbers (DIDs, "implicit" extensions) for the enterprise customer               | Can be dialed in different ways; The number configuration builds on top of the Pilot subscriber |
|           | PBX Group             | Forwards incoming calls to multiple extensions  | Ringling policy defines in which order the extensions will ring                                 |
| SIP Trunk | Digest authentication | Dynamically registers a remote IP PBX device  | Handles multiple users behind the IP PBX device   |
|           | IP authentication     | IP authentication of legacy IP PBX devices incapable of registering with the platform | Might require Trusted Subscriber and Trusted Source configuration                               |

---

**Tip**

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

---

## 5.7 SIP Peerings

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

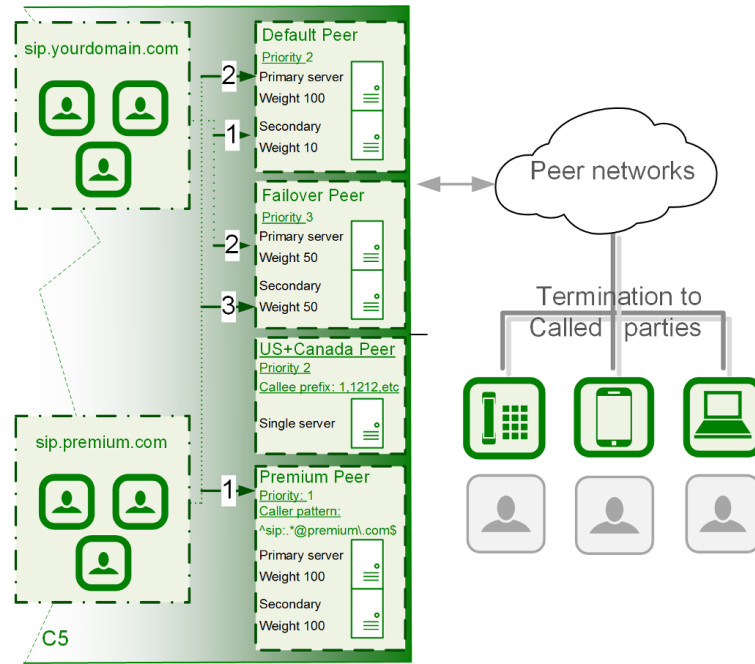
SIP peerings also enable incoming calls to your platform. For example, if you rent a pool of DID numbers from a SIP peer and offer them to your residential and business customers.

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

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

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

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



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

See the [Routing Order Selection](#) section for details about call routing.

Inbound rules allow [filtering out incoming INVITE requests](#) arriving from the corresponding SIP peering servers.

## 6 VoIP Service Configuration Scenario

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

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

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

---

### Tip

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

---

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

### 6.1 Creating a SIP Domain

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

---

### Tip

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

---

To create a SIP domain, follow these steps:

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

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

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

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

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

The screenshot shows the 'Create Domain' interface. At the top, there's a 'Reseller' section with a search bar. Below it is a table with the following data:

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

Below the table, it says 'Showing 1 to 2 of 2 entries'. There are navigation arrows and a '1' in a box. To the right is a 'Create Reseller' button. Below the table is a 'SIP Domain' input field containing 'sip.yourdomain.com'. At the bottom right is a 'Save' button.

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

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

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

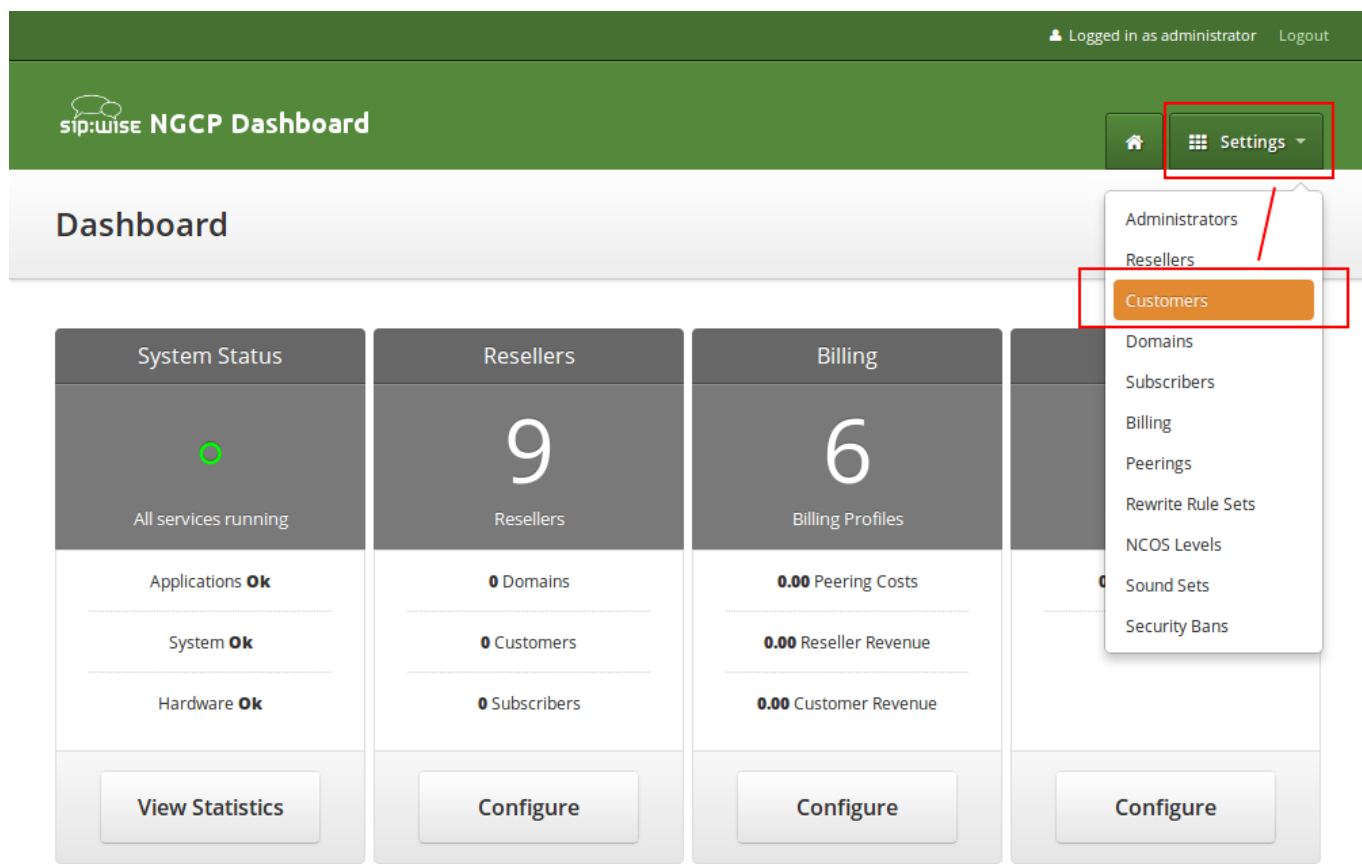
## 6.2 Creating a Customer

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

### Tip

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

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



Click on *Create Customer*.

Logged in as administrator Logout

sip:wise NGCP Dashboard

Home 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

© 2013 Sipwise GmbH, all rights reserved.

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

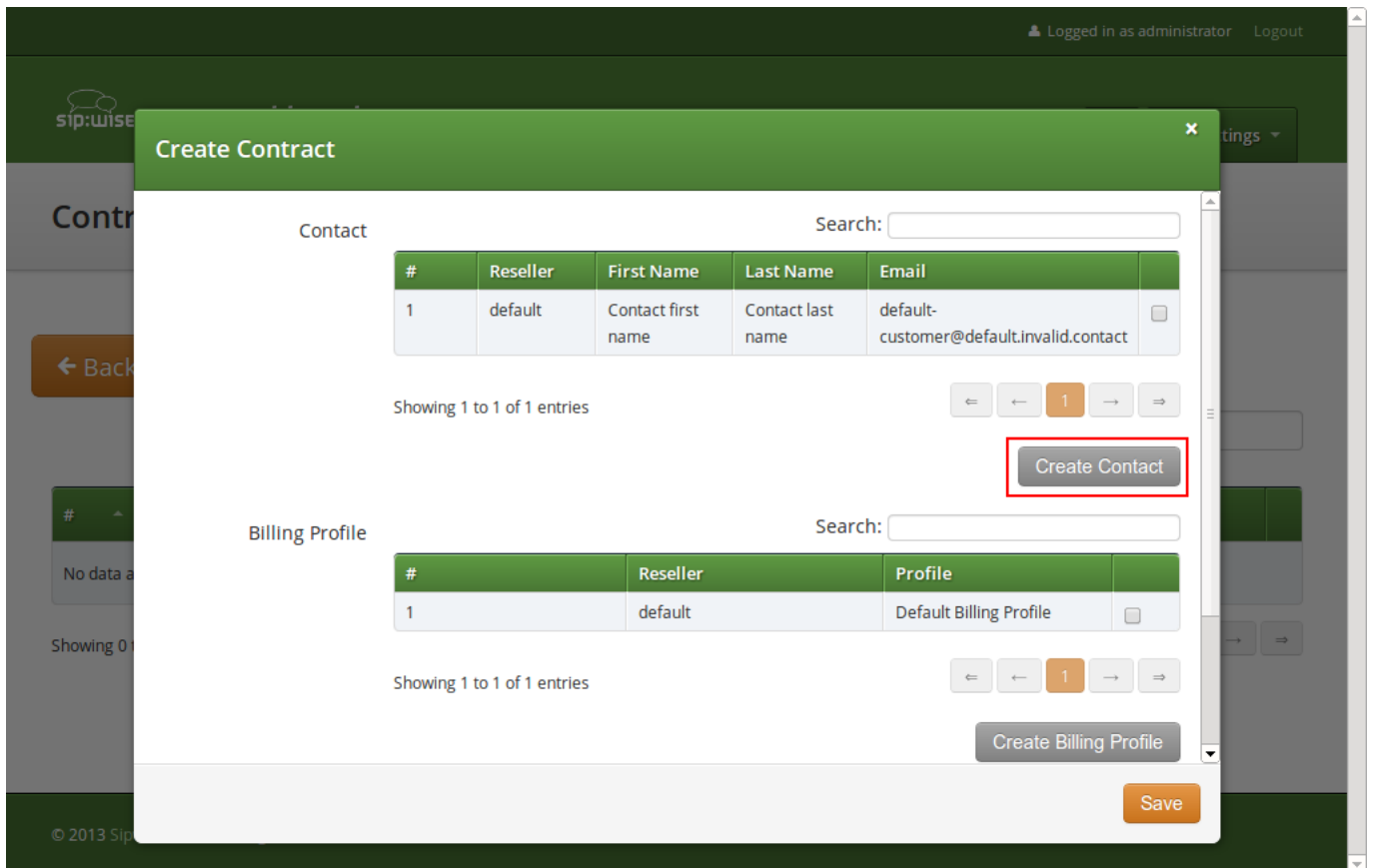
---

**Tip**

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

---

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



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



Logged in as administrator Logout

### Create Contact

Reseller Search:

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

Showing 1 to 1 of 1 entries

Create Reseller

First Name

Last Name

2 Email

Company

3 Save

© 2013 Sipwise

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

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

Logged in as administrator Logout

**sip:wise NGCP Dashboard** Home Settings

## Customers

Back Create Customer

Contract successfully created

Search:

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

Showing 1 to 1 of 1 entries

← 1 →

### 6.3 Creating a Subscriber

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

Logged in as administrator Logout

sip:wise NGCP Dashboard

Home Settings

## Customer Details

Back Edit

Reseller

Contact Details

Billing Profiles

1 Subscribers

★ Create Subscriber 2

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

Contract Balance

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

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

### Tip

This number can actually be used to place calls between local subscribers, even if you don't have any PSTN interconnection. This comes in handy if you use phones instead of soft-clients for your tests. The format in which this number can be dialled, so the subscriber is reached is defined in Section 6.7.

### Important



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

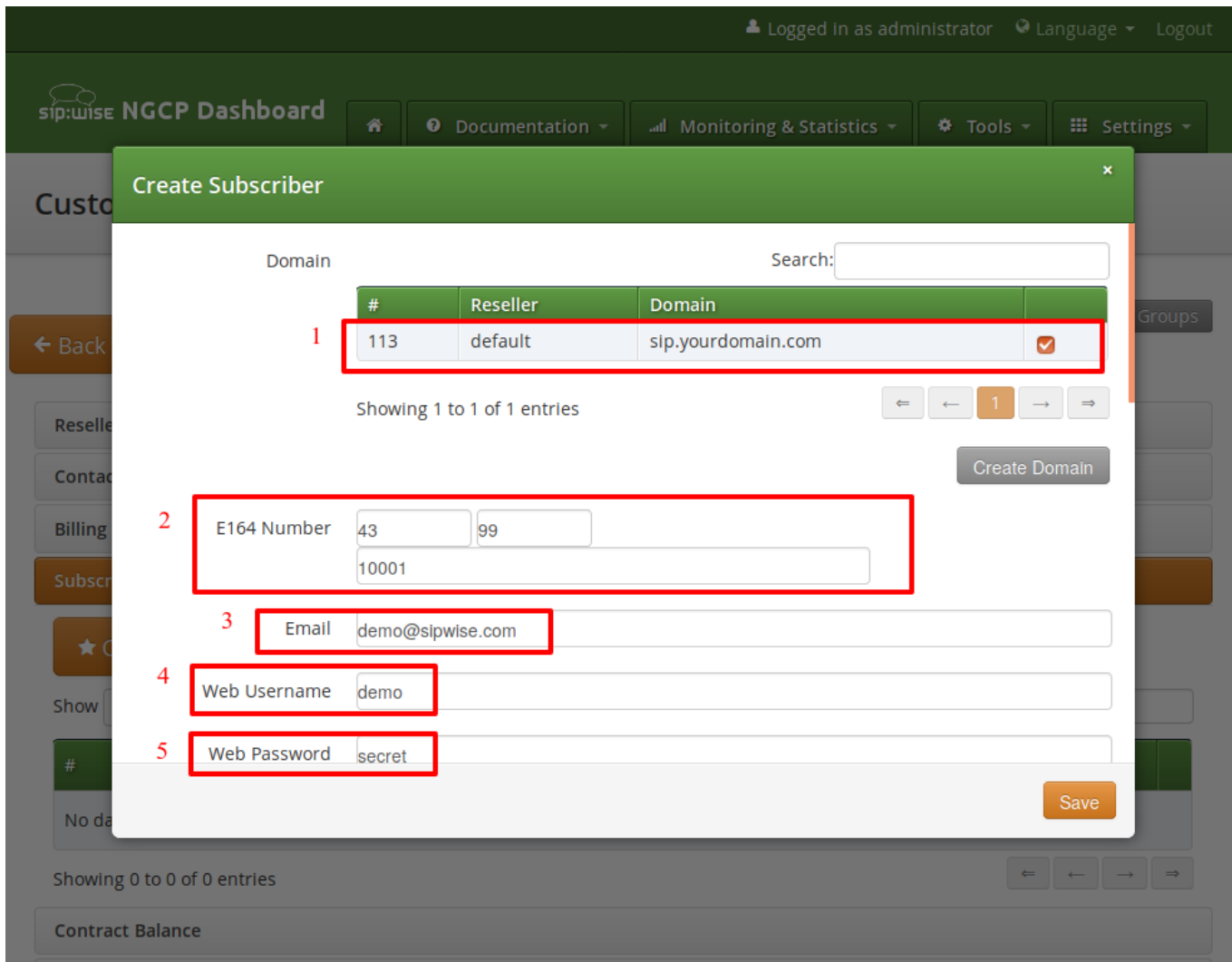
- **Email:** An email address for sending service-related notifications to.
- **Web Username:** This is the user part of the username the subscriber may use to log into her *Customer Self Care Interface*. The user part will be automatically suffixed by the SIP domain you choose for the **SIP URI**. Usually,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

**Caution**

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

---

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



The screenshot shows the 'Create Subscriber' form in the Sipwise NGCP Dashboard. The form is a modal window with a green header. It contains the following fields:

- Web Username: demo
- Web Password: secret
- SIP Username: demo (highlighted with a red box and labeled '6')
- SIP Password: 1secret! (highlighted with a red box and labeled '7')
- Lock Level: none
- Status: active
- External ID: demo
- Administrative:

A 'Save' button is located at the bottom right of the form. The background shows the dashboard navigation menu with options like 'Documentation', 'Monitoring & Statistics', 'Tools', and 'Settings'.

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

---

**Tip**

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

---

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

## 6.4 Domain Preferences

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

Logged In as administrator Language Logout

sip:wise NGCP Dashboard Home Documentation Monitoring & Statistics Tools Settings

## Domains

← Back ★ Create Domain

Show 5 entries Search:

| #   | Reseller | Domain             |                    |
|-----|----------|--------------------|--------------------|
| 113 | default  | sip.yourdomain.com | Delete Preferences |

Showing 1 to 1 of 1 entries

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 6.7, then assign it to the domain by editing the *rewrite\_rule\_set* preference.







## Domain "sip.yourdomain.com" - Preferences

← Back

Call Blockings

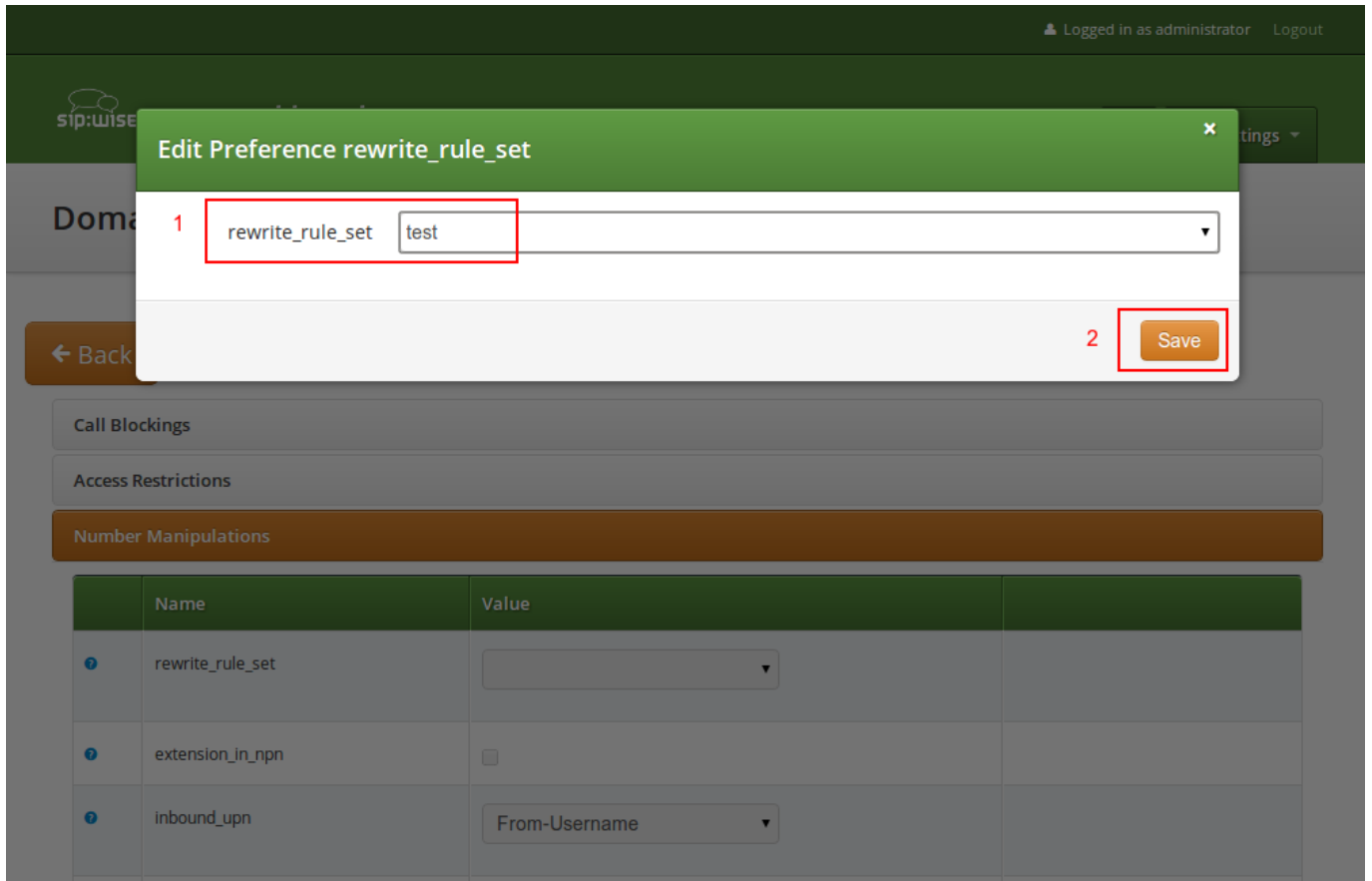
Access Restrictions

1 Number Manipulations

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

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





Then, select the field you want the *User Provided Number* to be taken from for inbound INVITE messages. Usually the *From-Username* should be fine, but you can also take it from the *Display-Name* of the From-Header, and other options are available as well.

## 6.5 Subscriber Preferences

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

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

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

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

**Note**

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

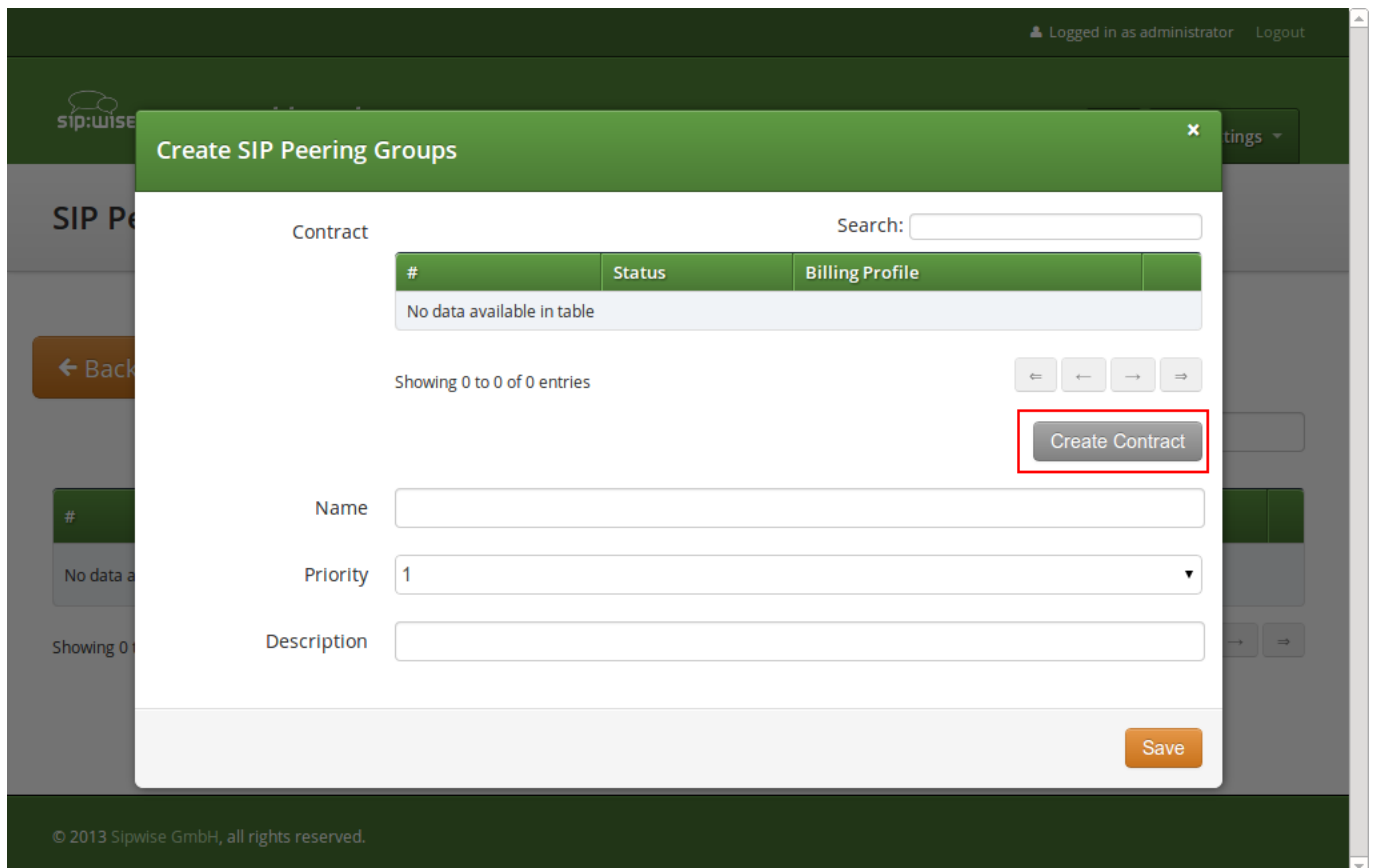
## 6.6 Creating Peerings

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

### 6.6.1 Creating Peering Groups

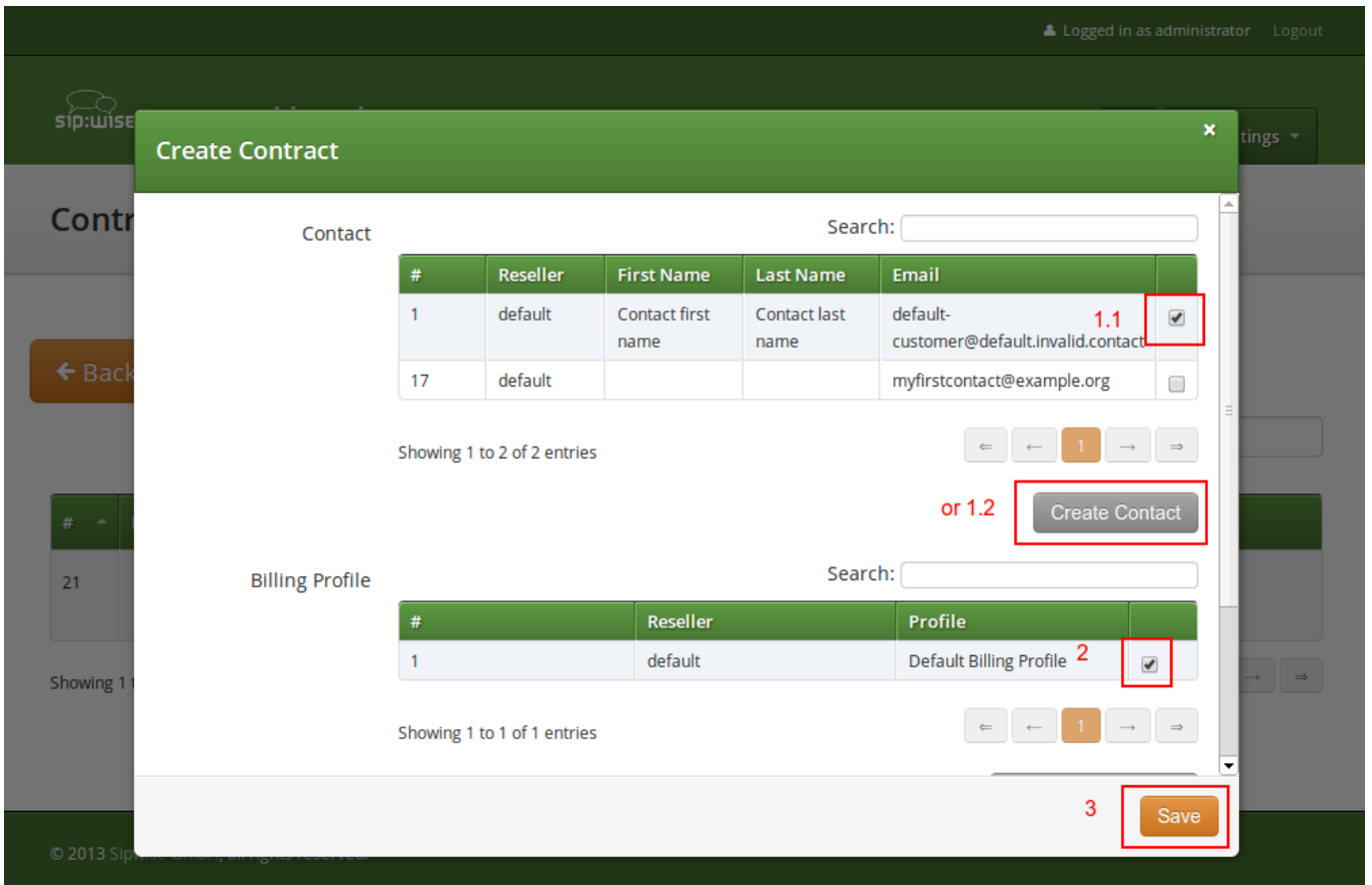
Click on *Create Peering Group* to create a new group.

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



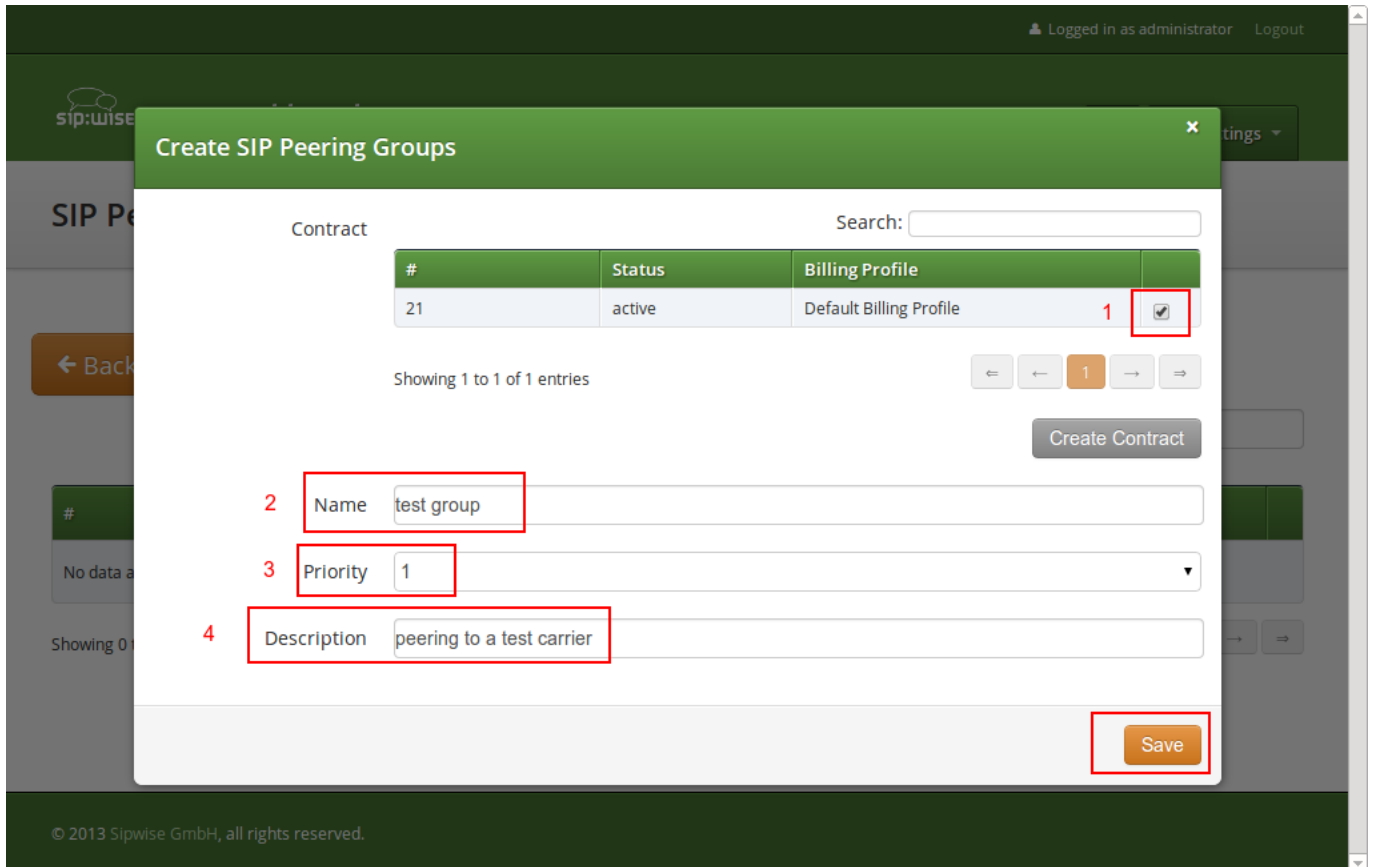
The screenshot shows the 'Create SIP Peering Groups' dialog box in the Sipwise web interface. The dialog is titled 'Create SIP Peering Groups' and has a search bar for 'Contract'. Below the search bar is a table with columns '#', 'Status', and 'Billing Profile'. The table is currently empty, displaying 'No data available in table'. Below the table, it says 'Showing 0 to 0 of 0 entries'. There are navigation buttons (back, forward) and a 'Create Contract' button highlighted with a red box. Below the table are form fields for 'Name', 'Priority' (set to 1), and 'Description'. A 'Save' button is located at the bottom right of the dialog. The background shows the Sipwise interface with a 'Logged in as administrator' status and a 'Logout' link.

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



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

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



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

Then click *Save* to create the group.

## 6.6.2 Creating Peering Servers

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

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

**Peering Servers**

← Back
★ Create Peering Server

Show 5 entries Search:

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

Showing 0 to 0 of 0 entries ← ← → →

**Outbound Peering Rules**

ANY of the rules must match to choose the peering group for outbound calls.

★ Create Outbound Peering Rule

Show 5 entries Search:

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

Showing 0 to 0 of 0 entries ← ← → →

**Inbound Peering Rules**

ALL of the rules must match to choose the peering group for inbound calls.

★ Create Inbound Peering Rule

Show 5 entries Search:

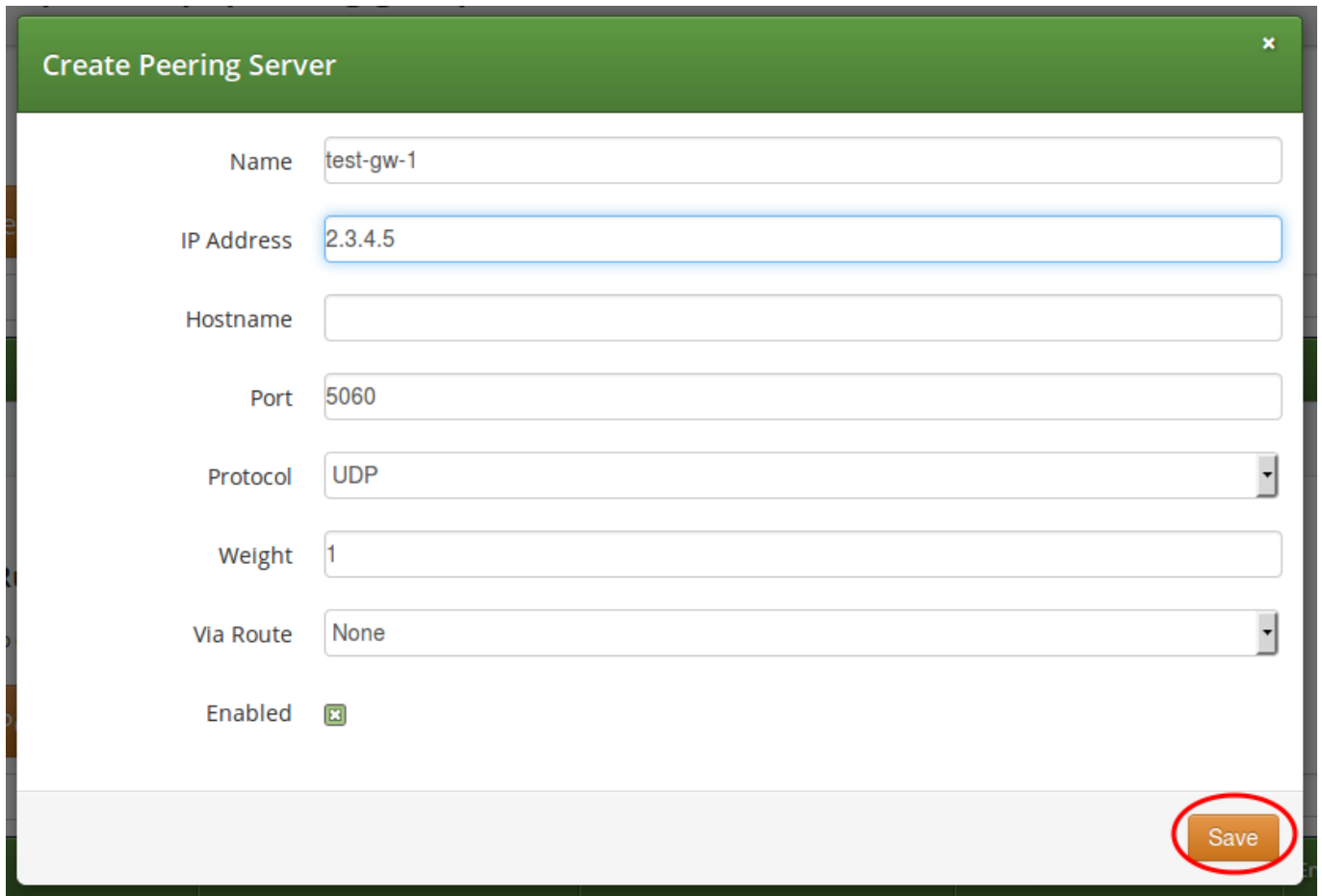
| Priority                   | # | Field | Pattern | Reject Code | Reject Reason | Enabled |
|----------------------------|---|-------|---------|-------------|---------------|---------|
| No data available in table |   |       |         |             |               |         |

Showing 0 to 0 of 0 entries ← ← → →

Figure 9: Create Peering Server

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

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



The screenshot shows a web form titled "Create Peering Server" with a green header. The form contains the following fields and values:

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

A "Save" button is located at the bottom right of the form, circled in red.

Figure 10: Peering Server Properties

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

---

**Tip**

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

---

**Tip**

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

---

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

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.

### 6.6.2.1 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](#)
[★ Create Peering Server](#)

Peering server successfully created

Show 5 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       |

Showing 1 to 1 of 1 entries ← 1 →

**Outbound Peering Rules**

ANY of the rules must match to choose the peering group for outbound calls.

[★ Create Outbound Peering Rule](#)

Show 5 entries Search:

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

Showing 0 to 0 of 0 entries ← →

**Inbound Peering Rules**

ALL of the rules must match to choose the peering group for inbound calls.

[★ Create Inbound Peering Rule](#)

Figure 11: Create Outbound Peering Rule

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

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



The screenshot shows a 'Create Outbound Peering Rule' dialog box. It features a green title bar with a close button. Below the title bar are five input fields: 'Callee prefix', 'Callee pattern', 'Caller pattern', 'Description' (with the text 'Default rule'), and 'Enabled' (with a checked checkbox). A red oval highlights the 'Description' field and the 'Save' button at the bottom right.

Figure 12: Outbound Peering Rule Properties

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

---

**Tip**

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

---

**Tip**

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

---

**Tip**

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

---

### 6.6.2.2 Inbound Peering Rules

Starting from *mr5.0* release, Sipwise C5 supports filtering SIP INVITE requests sent by SIP peers. The system administrator may define one or more matching rules for SIP URIs that are present in the headers of SIP INVITE requests, and select which SIP header (or part of the header) must match the pattern declared in the rule.

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



**Caution**

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

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

**Peering Servers**

← Back    ★ Create Peering Server

Show 5 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       |

Showing 1 to 1 of 1 entries

**Outbound Peering Rules**

ANY of the rules must match to choose the peering group for outbound calls.

★ Create Outbound Peering Rule

Show 5 entries    Search:

| # | Callee Prefix | Callee Pattern | Caller Pattern | Description  | Enabled |
|---|---------------|----------------|----------------|--------------|---------|
| 1 |               |                |                | Default rule | 1       |

Showing 1 to 1 of 1 entries

**Inbound Peering Rules**

ALL of the rules must match to choose the peering group for inbound calls.

★ Create Inbound Peering Rule

Show 5 entries    Search:

| Priority                   | # | Field | Pattern | Reject Code | Reject Reason | Enabled |
|----------------------------|---|-------|---------|-------------|---------------|---------|
| No data available in table |   |       |         |             |               |         |

Showing 0 to 0 of 0 entries

Figure 13: Create Inbound Peering Rule

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

Figure 14: 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.

**Tip**

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

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

**Peering Servers**

← Back    ★ Create Peering Server

Show 5 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       |

Showing 1 to 1 of 1 entries

← ← 1 → →

**Outbound Peering Rules**

ANY of the rules must match to choose the peering group for outbound calls.

★ Create Outbound Peering Rule

Show 5 entries    Search:

| # | Callee Prefix | Callee Pattern | Caller Pattern | Description  | Enabled |
|---|---------------|----------------|----------------|--------------|---------|
| 1 |               |                |                | Default rule | 1       |

Showing 1 to 1 of 1 entries

← ← 1 → →

**Inbound Peering Rules**

ALL of the rules must match to choose the peering group for inbound calls.

★ Create Inbound Peering Rule

Show 5 entries    Search:

| Priority | # | Field     | Pattern     | Reject Code | Reject Reason               | Enabled |
|----------|---|-----------|-------------|-------------|-----------------------------|---------|
| 50       | 1 | to_domain | example\org | 403         | Invalid called party domain | 1       |

Showing 1 to 1 of 1 entries

← ← 1 → →

Figure 15: Peering Servers Overview

### 6.6.2.3 Routing Order Selection

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

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

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



#### Important

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

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

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

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



#### Important

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

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

```
#!/usr/bin/perl

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

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

my $iters = 10000;
my @rands;

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

```

}

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

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

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

```

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

```

# lcr_weight_test.pl 1 2

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

```

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

---

#### Note

The Sipwise C5 may use a slightly different approach in selecting the appropriate peering server if the *peer probing* feature is enabled. See the details in Section 7.12 of the handbook.

---

### 6.6.2.4 Least Cost Routing (LCR) Configuration

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

Sipwise C5 also allows you routing calls to the cheapest SIP peers saving your termination cost.

To enable LCR routing, do the following:

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

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

1. All peering groups that meet the following criteria configured in the outbound peering rule are added to the list of routes for a particular call (for pure LCR you might want to omit these filters leaving them blank):
  - Callee's username matches *callee prefix*
  - Callee's URI matches *callee pattern*
  - Caller's URI matches *caller pattern*
2. When all matching peering groups are selected, the longest matching *callee prefix* is selected from each of them. And the peering groups are *temporary* ordered according to the longest matching prefix and priority.
3. Then, the LCR module re-orders the peering groups starting from the lowest termination cost to the highest (ignoring the prefix length and peering group priorities).
4. The platform will first route the call to the servers of the first peering group in this list. If no peering server can terminate the call, the call would fail-over to the second peering group from the list and so on.

---

**Note**

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

---

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

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

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

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

## 6.6.3 Authenticating and Registering against Peering Servers

### 6.6.3.1 Proxy-Authentication for outbound calls

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

Peering Servers

← Back   ★ Create Peering Server

Show 5 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       | <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Preferences</a> |

Showing 1 to 1 of 1 entries

Outbound Peering Rules

ANY of the rules must match to choose the peering group for outbound calls.

★ Create Outbound Peering Rule

Show 5 entries   Search:

| # | Callee Prefix | Callee Pattern | Caller Pattern | Description  | Enabled |  |
|---|---------------|----------------|----------------|--------------|---------|--|
| 1 |               |                |                | Default rule | 1       |  |

Showing 1 to 1 of 1 entries

Inbound Peering Rules

ALL of the rules must match to choose the peering group for inbound calls.

★ Create Inbound Peering Rule

Figure 16: Select Peering Server Preferences

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

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



[← Back](#)


Preference peer\_auth\_realm successfully updated.

Access Restrictions

Number Manipulations

NAT and Media Flow Control

Remote Authentication

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

Session Timers

### Important



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

### Tip

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

## 6.6.3.2 Registering at a Peering Server

Unfortunately, the credentials configured above are not yet automatically used to register Sipwise C5 at your peer hosts. There is however an easy manual way to do so, until this is addressed.

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

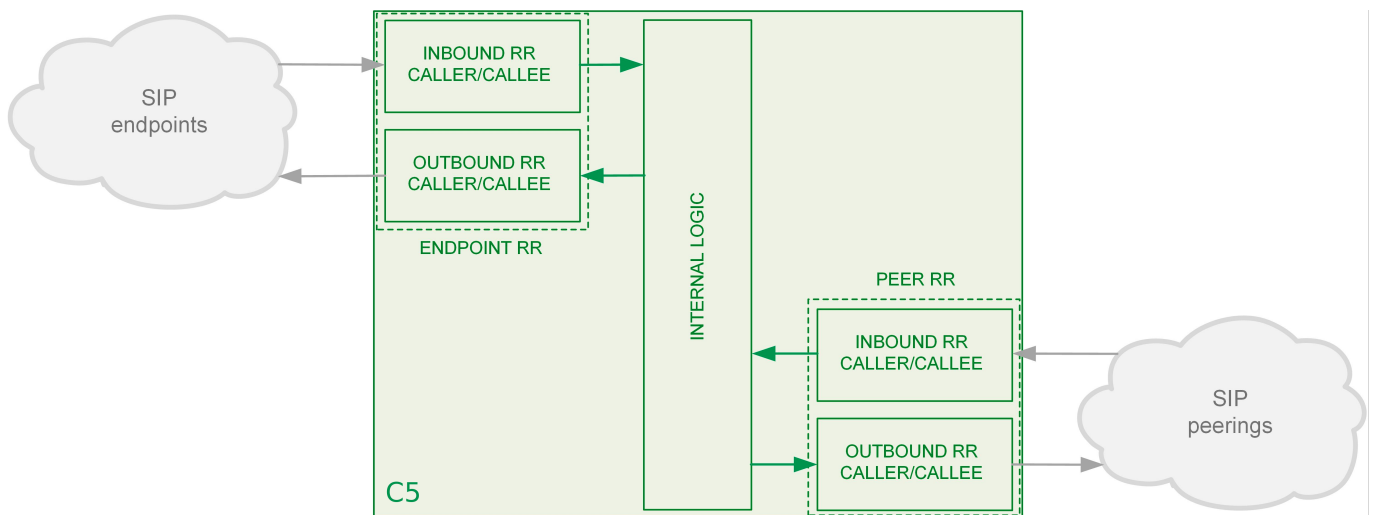


### Important

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

## 6.7 Configuring Rewrite Rule Sets

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



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

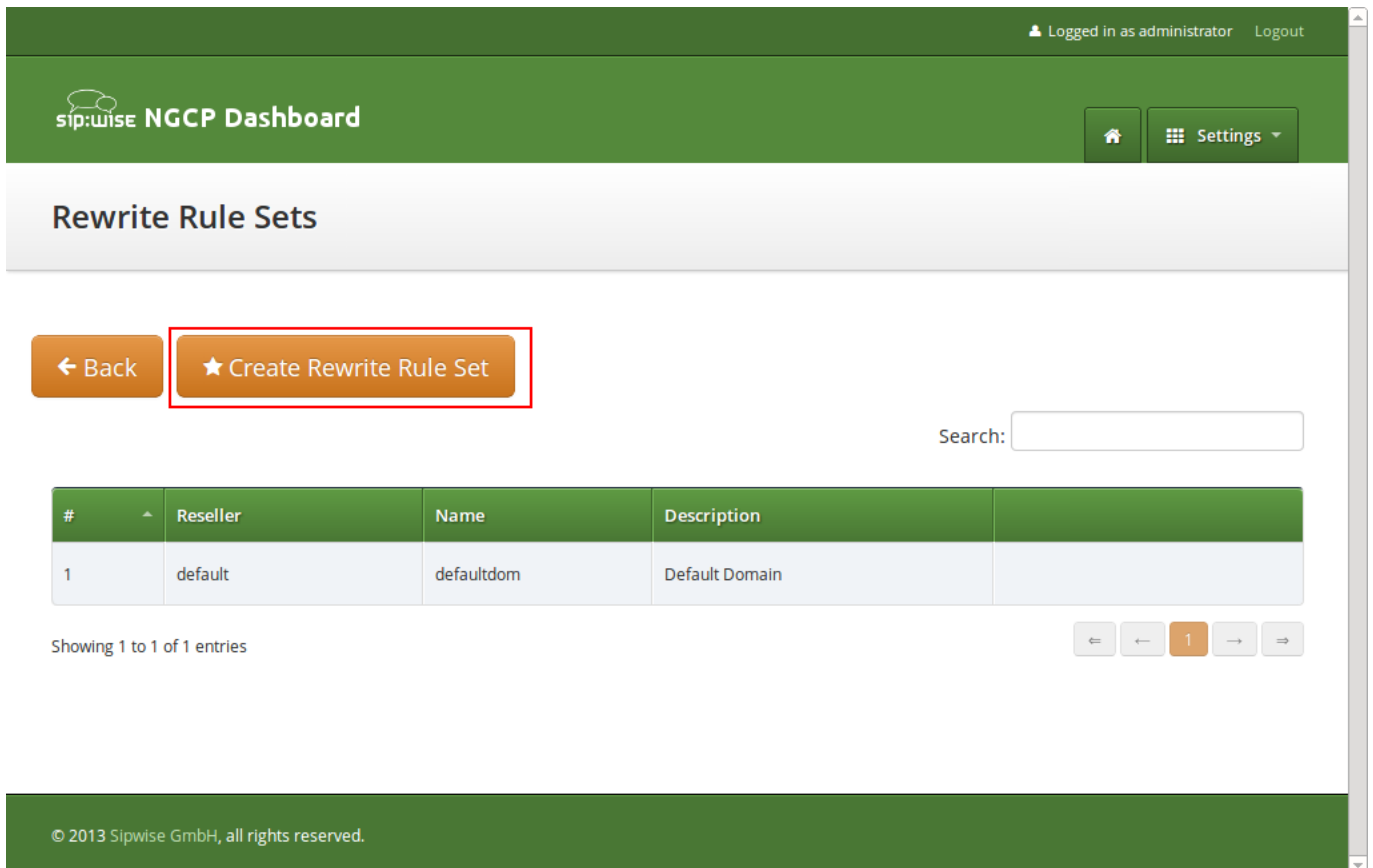
- Call from local subscriber A to local subscriber B: Inbound RR from local Domain/Subscriber A and Outbound Rewrite Rules from local Domain/Subscriber B.
- Call from local subscriber A to the peer: Inbound RR from local Domain/Subscriber A and Outbound Rewrite Rules from the peer.
- Call from peer to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

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

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

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

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



The screenshot shows the NGCP Dashboard interface. At the top, it says "Logged in as administrator" and "Logout". The dashboard title is "sip:wise NGCP Dashboard". Below the title, there are navigation buttons for "Home" and "Settings". The main heading is "Rewrite Rule Sets".

Below the heading, there are two buttons: "Back" and "Create Rewrite Rule Set". The "Create Rewrite Rule Set" button is highlighted with a red box. To the right of these buttons is a search input field labeled "Search:".

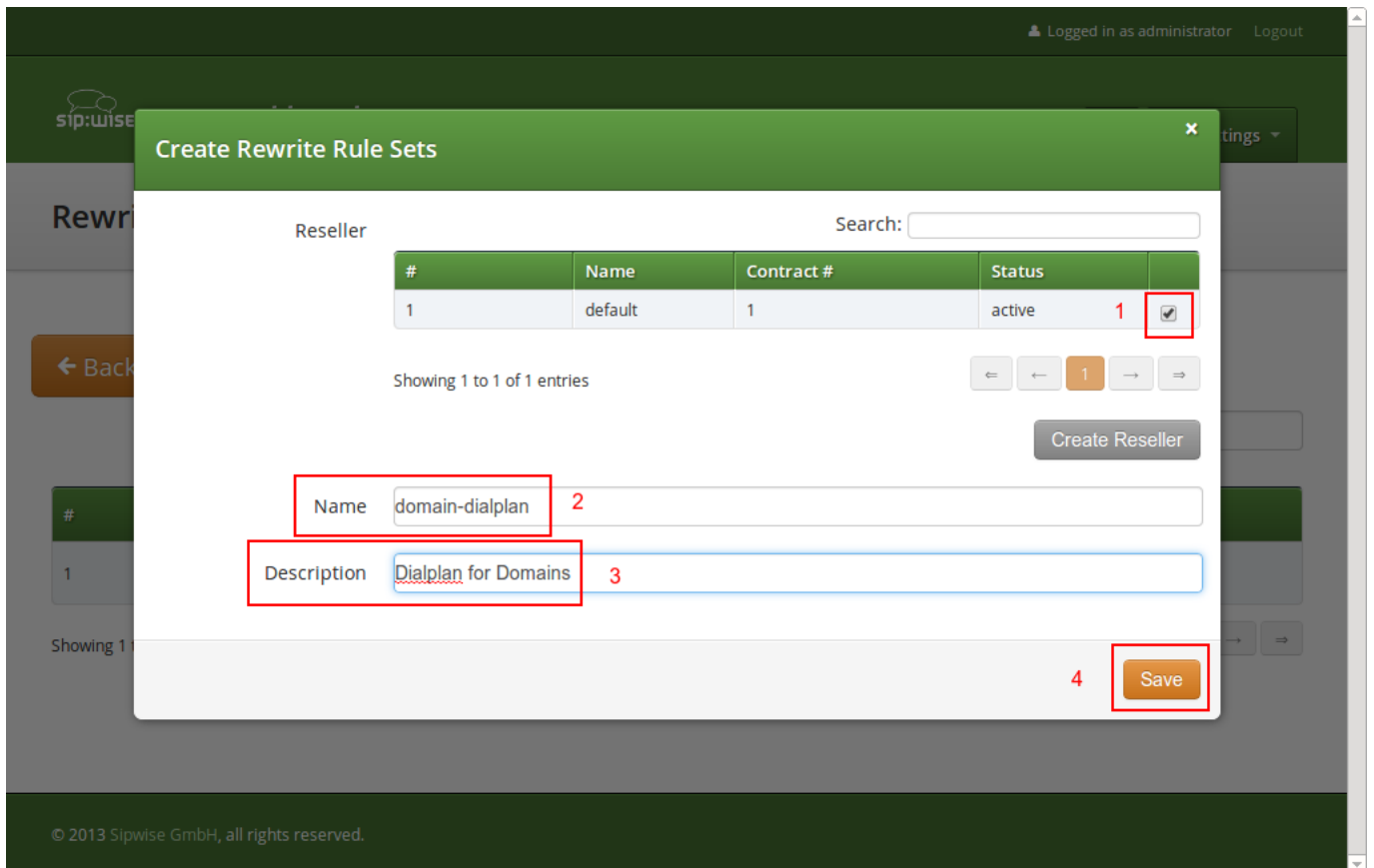
Below the search field is a table with the following data:

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

Below the table, it says "Showing 1 to 1 of 1 entries". There are also pagination controls showing "1" in a highlighted orange box.

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

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



Press the *Save* button to create the set.

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

Logged in as administrator Logout

sip:wise NGCP Dashboard

Home Settings

## Rewrite Rule Sets

Back Create Rewrite Rule Set

Rewrite rule set successfully created

Search:

| # | Reseller | Name            | Description          |   |
|---|----------|-----------------|----------------------|---|
| 1 | default  | defaultdom      | Default Domain       |   |
| 2 | default  | domain-dialplan | Dialplan for Domains | <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Rules</a> |

Showing 1 to 2 of 2 entries

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

### Tip

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

#### 6.7.1 Inbound Rewrite Rules for Caller

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

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

STRIP LEADING 00 OR +

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

- Field: Caller

REPLACE 0 BY CALLER'S COUNTRY CODE:

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

NORMALIZE LOCAL CALLS:

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

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

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



**Important**

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

**Tip**

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

## Rewrite Rules for domain-dialplan

← Back

★ Create Rewrite Rule

Rewrite rule successfully created

Inbound Rewrite Rules for Caller

|   | Match Pattern   | Replacement Pattern     | Description                                  |
|---|---|-------------------------|--|
| 1 | ↑ ↓ <input type="checkbox"/> <input type="checkbox"/>   | ^(00 \+)([1-9][0-9]+)\$ | \2<br>International to E.164                 |
|   | ↑ ↓ <input type="checkbox"/> <input type="checkbox"/> 2 | ^0([1-9][0-9]+)\$       | #{caller_cc}\1<br>National to E.164          |
|   | ↑ ↓ <input type="checkbox"/> <input type="checkbox"/>   | ^([1-9][0-9]+)\$        | #{caller_cc}#{caller_ac}\1<br>Local to E.164 |

Inbound Rewrite Rules for Callee

Outbound Rewrite Rules for Caller

Outbound Rewrite Rules for Callee

### 6.7.2 Inbound Rewrite Rules for Callee

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

STRIP LEADING 00 OR +

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

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

REPLACE 0 BY CALLER'S COUNTRY CODE:

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

NORMALIZE AREACODE-INTERNAL CALLS:

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

---

### Tip

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

---

### 6.7.3 Outbound Rewrite Rules for Caller

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

REPLACE AUSTRIAN COUNTRY CODE 43 BY 0

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



- **Direction:** Outbound
- **Field:** Caller

#### PREFIX 00 FOR INTERNATIONAL CALLER

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

---

#### Tip

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

---

### 6.7.4 Outbound Rewrite Rules for Callee

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

#### PREFIX SIPSP# FOR ALL CALLS

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

### 6.7.5 Emergency Number Handling

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

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

### 6.7.5.1 Simple Emergency Number Handling Overview

The overview of emergency call processing is as follows:

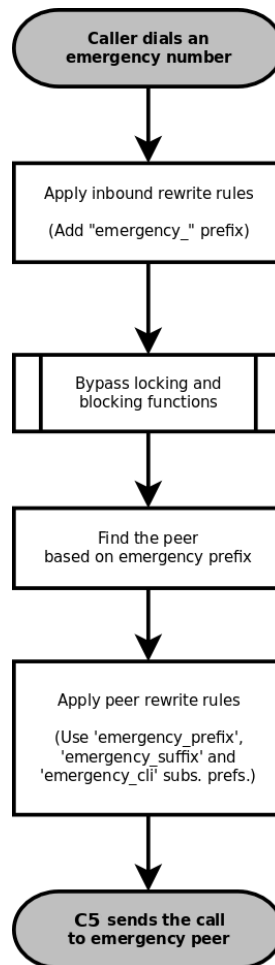


Figure 17: Simple Emergency Call Handling

Configuring Emergency Numbers is also done via Rewrite Rules.

### 6.7.5.2 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.

### 6.7.5.3 Normalize Emergency Calls for Peers

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

#### NORMALIZE EMERGENCY CALL FOR PEER

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

### 6.7.6 Assigning Rewrite Rule Sets to Domains and Subscribers

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

← Back

Call Blockings

Access Restrictions 1

Number Manipulations

|   | Name               | Value                    |        |
|---|--------------------|--------------------------|--------|
| ? | rewrite_rule_set 2 | defaultdom               | 3 Edit |
| ? | extension_in_npn   | <input type="checkbox"/> |        |
| ? | inbound_upn        | From-Username            |        |
| ? | outbound_from_user | User-Provided-Number     |        |

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

### 6.7.7 Creating Dialplans for Peering Servers

For each peering server, you can use one of the Rewrite Rule Sets that was created previously as explained in Section 6.7 (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.

### 6.7.8 Call Routing Verification

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

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

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

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

### **Expected input data**

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

### **Examples**

1. Using only phone numbers and explicit subscriber selection

- Input Data:

### Call Routing Verification

[← Back](#) [Expand Groups](#)

Caller number/url:

Callee number/url:

Caller Type:  Subscriber  Peer

Caller Subscriber Search:

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

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

Caller Rewrite Rules Override

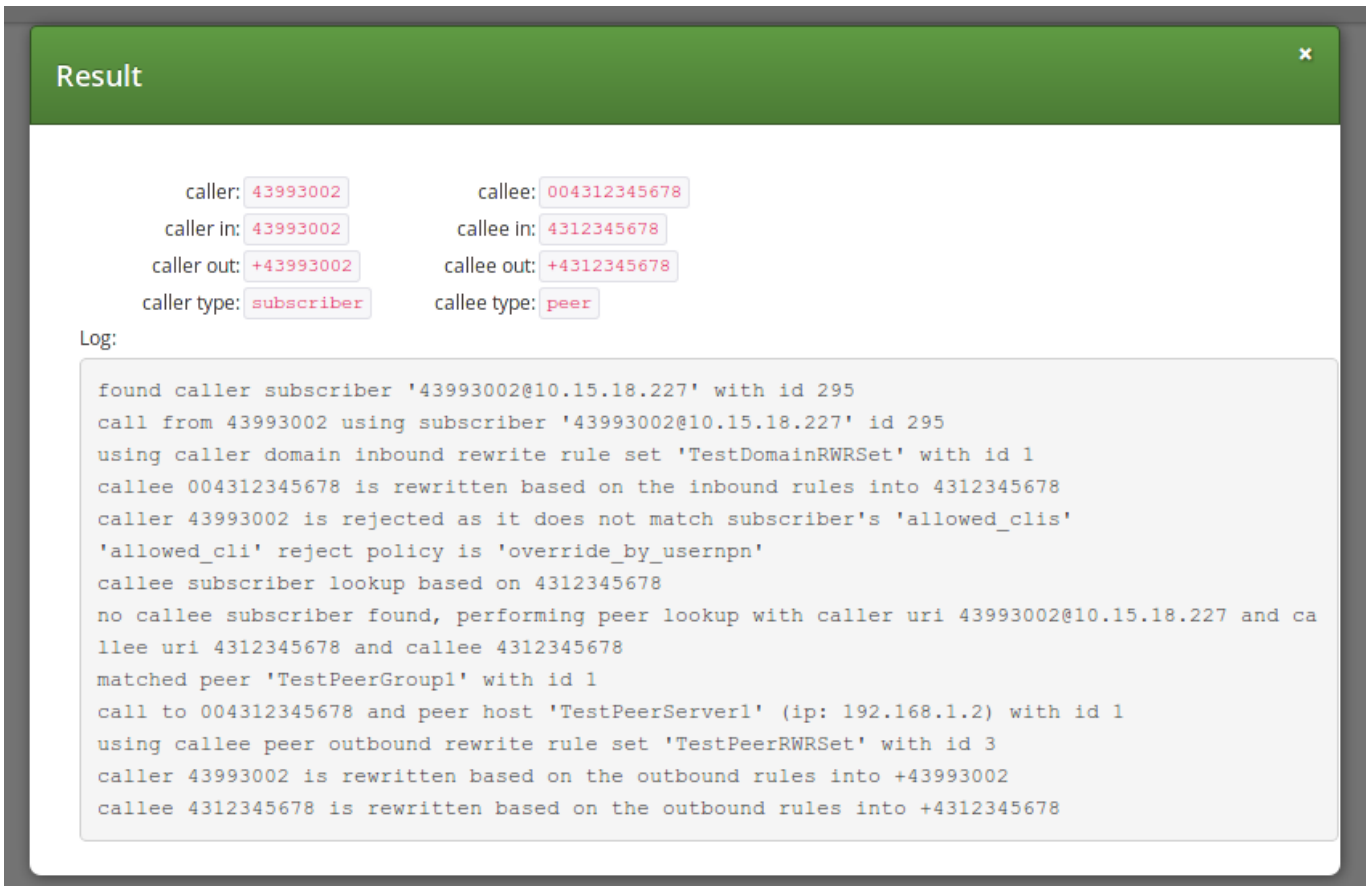
Callee Rewrite Rules Override

Callee Peer Override

[Verify](#)

Figure 18: Call Routing Verif. - Only Numbers - Input

- Result:



The screenshot shows a window titled "Result" with a green header and a close button. Below the header, there are two columns of call details:

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

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

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

Figure 19: Call Routing Verif. - Only Numbers - Result

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

- Input Data:

### Call Routing Verification

[← Back](#) [Expand Groups](#)

Caller number/uri:

Callee number/uri:

Caller Type:  Subscriber  Peer

Caller Subscriber Search:

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

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

Caller Rewrite Rules Override

Callee Rewrite Rules Override

Callee Peer Override

[Verify](#)

Figure 20: Call Routing Verif. - Number and URI - Input

- Result:



### Result

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

Log:

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

Figure 21: Call Routing Verif. - Number and URI - Result

## 7 Features

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

### 7.1 About the Admin Web Interface

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

#### 7.1.1 Filtering the Lists / Datables

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

The screenshot shows the 'Subscribers' page in the Admin Web Interface. At the top left, there is a 'Back' button. Below it, a 'Show' dropdown is set to '10' entries. A search bar on the right contains the text '200'. The main area displays a table with 5 rows of subscriber data. At the bottom left, a status bar indicates 'Showing 1 to 5 of 5 entries (filtered from 49 total entries)'. At the bottom right, there are pagination controls showing '1' of 1 pages.

| #   | Contract # | Contact Email                   | Username          | Domain              | UUID                                 | Status | Number       | Profile |
|-----|------------|---------------------------------|-------------------|---------------------|--------------------------------------|--------|--------------|---------|
| 39  | 3          | tv@enterprise.org               | ext200            | 1.c5.████████.com   | f25dc6e-c56e-431a-9e7a-eada925f0de7  | active | 40333100200  |         |
| 105 | 21         | machsols4b_customer@example.org | machsols4b_sub2   | ████████.com        | afa78b9d-c02b-41de-9a77-d82006698e7a | active | 438882200102 |         |
| 121 | 27         | basicsip@boghici.au             | 555200            | c5.████████.com     | 42e03f31-c068-46d8-9a58-0f2f9d938749 | active | 39555200     |         |
| 159 | 19         | cbs@tt.org                      | ext200            | c5.████████.com     | 86195f25-ab71-4aaa-85a8-caefb33b3fc0 | active | 44266200     |         |
| 35  | 21         | machsols4b_customer@example.org | machsols4b_sub101 | s4b.c5.████████.com | 377394fe-2f67-4feb-8013-3fc35544807a | active | 438881200101 |         |

Figure 22: Filtered List of Subscribers

#### The Search String

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

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

**Note**

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

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

**7.1.2 Call History**

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

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

Call List for machsols4b\_subs2@ [redacted] (43 888 2200102)

← Back

Show all calls

Show 10 entries From Date: To Date: Search: s4b\_int\*

| #     | Caller       | Callee                         | CLIR | Billing zone | Status   | Start Time              | Duration    | MOS avg | MOS packetloss | MOS jitter | MOS roundtrip | Call-ID                           | Cost |
|-------|--------------|--------------------------------|------|--------------|----------|-------------------------|-------------|---------|----------------|------------|---------------|-----------------------------------|------|
| 1505  | 438882200102 | s4b_int432158@ [redacted]      | 0    |              | cancel   | 2018-08-06 11:27:41.945 | 0:00:00     |         |                |            |               | d54aec4a71e57db7c38db9e8c8f894e1d | 0.00 |
| 1507  | 438882200102 | s4b_int31882200101@ [redacted] | 0    |              | noanswer | 2018-08-06 11:28:10.273 | 0:00:00     |         |                |            |               | d2f5a87577ee338fb326743fb2894e1d  | 0.00 |
| 1509  | 438882200102 | s4b_int31882200101@ [redacted] | 0    | all          | ok       | 2018-08-06 11:29:54.920 | 0:00:21.891 | 4.3     | 0.0            | 0.0        | 9999.0        | 13148ad1d3c16da8eba7dd5443894e1d  | 0.00 |
| Total |              |                                |      |              |          |                         | 0:00:21.891 |         |                |            |               |                                   | 0.00 |

Showing 1 to 3 of 3 entries (filtered from 10 total entries)

Figure 23: Filtered Call History



**Caution**

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

## 7.2 Managing System Administrators

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

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

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

### 7.2.1 Configuring Administrators

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

The screenshot shows the 'Administrators' management page. At the top, there are two buttons: '← Back' and '★ Create Administrator'. The 'Create Administrator' button is circled in red. Below the buttons, a light blue message bar states 'Administrator successfully updated'. Underneath, there is a search bar and a 'Show 5 entries' dropdown. The main content is a table with the following columns: #, Reseller, Login, Master, Active, Read Only, Show Passwords, Show CDRs, Show Billing Info, Lawful Intercept, and a set of action buttons (Edit, Delete, API key). The 'Edit' button for the 'Demo Reseller' row is circled in red. At the bottom, there is a pagination control showing 'Showing 1 to 2 of 2 entries' and navigation arrows.

| # | 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                | <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">API key</a> |

Figure 24: 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 |                                     |
|----|---------------|------------|--------|-------------------------------------|
| 1  | default       | 1          | active | <input checked="" type="checkbox"/> |
| 61 | Demo Reseller | 243        | active | <input type="checkbox"/>            |

Showing 1 to 2 of 2 entries

Login

Password

Email

Is superuser

Figure 25: Generic System Administrator Attributes

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

**Note**

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

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

### 7.2.2 Access Rights of Administrators

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

The screenshot shows a web interface titled "Edit Administrator" with a close button (X) in the top right corner. The form contains the following access rights, each with a checkbox:

- Is superuser
- Is master
- Is ccare
- Is active
- Read only
- Show passwords
- Call data
- Billing data
- Lawful intercept
- Can reset password
- Is system

A "Save" button is located at the bottom right of the form.

Figure 26: Access Rights of System Administrators

Table 2: Access Rights of System Administrators

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

Table 2: (continued)

| Label in admin list | Access Right   | Description   |
|---------------------|----------------|---|
| Customer Care       | Is ccare       | The user is allowed to create, delete or modify Customers and Subscribers. If mixed with <b>Is superuser</b> it defines whether the user can modify the data only within a Reseller the user belongs to, or across all Resellers. The user can access to relevant information required to create or modify Customers or Subscribers, such as Domains, Billing Profiles, Email Templates, but the user cannot access the entries (e.g: see the detailed info about a Billing Profile), nor modify them.  |
| Active              | Is active      | The user account is active, i.e. the admin user can login on the web panel or authenticate himself on REST API; otherwise user authentication will fail.  |
| Read Only           | Read only      | The user will only be able to list various data but is not allowed to modify anything. <ul style="list-style-type: none"> <li>• For the <b>web interface</b> this means that <i>Create...</i> and <i>Edit</i> buttons will be hidden or disabled.</li> <li>• For the <b>REST API</b> this means that only <code>GET</code>, <code>HEAD</code>, <code>OPTIONS</code> HTTP request methods are accepted, and Sipwise C5 will reject those targeting data modification: <code>PUT</code>, <code>PATCH</code>, <code>POST</code>, <code>DELETE</code>.</li> </ul> |
| Show Passwords      | Show passwords | The user sees subscriber passwords (in plain text) on the web interface. <hr/> <p><b>Note</b></p> Admin panel user passwords and subscriber web passwords are stored in an unreadable way (cryptographic hash digest) in the database, while subscriber SIP passwords are stored in plain text. The latter happens on purpose, e.g. to make subscriber data migration possible. <hr/>   |
| Show CDRs           | Call data      | This privilege has effect on 2 items that will be displayed on admin panel of NGCP, when <i>Subscriber</i> → <i>Details</i> is selected: <ol style="list-style-type: none"> <li>1. <i>PBX Groups</i> list</li> <li>2. <i>Captured Dialogs</i> list</li> </ol>   |
| Show Billing Info   | Billing data   | Some REST API resources that are related to billing are disabled: HTTP requests on <code>/api/vouchers</code> , <code>/api/topupcash</code> and <code>/api/topupvoucher</code> resources are rejected.  |

Table 2: (continued)

| Label in admin list | Access Right          | Description  |
|---------------------|-----------------------|--|
| Lawful Intercept    | Lawful<br>intercept   | Visible only for System administrators. If the privilege is selected then the REST API for interceptions (that is: <code>/api/interceptions</code> ) is enabled; if the privilege is not selected then the interceptions API is disabled. Administrators with this flag do not have access to anything but the administrators in UI and API and <code>/api/interceptions</code> in the API<br><br><b>Note</b><br>This means that besides enabling LI in <code>config.yml</code> configuration file one also needs to enable the API via the LI privilege of an administrator user, so that Sipwise C5 can really provide LI service. |
| Can Reset Password  | Can Reset<br>Password | The user is allowed to request a password reset.   |
| System              | Is system             | The user is allowed to create, delete or modify Lawful Intercept Admins which are otherwise invisible to other types of administrators.  |

### 7.3 Access Control for SIP Calls

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

#### 7.3.1 Block Lists

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



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

Block Lists are separated into *Administrative Block Lists* (*adm\_block\_\**) and *Subscriber Block Lists* (*block\_\**). They both have the same 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.

### 7.3.1.1 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.

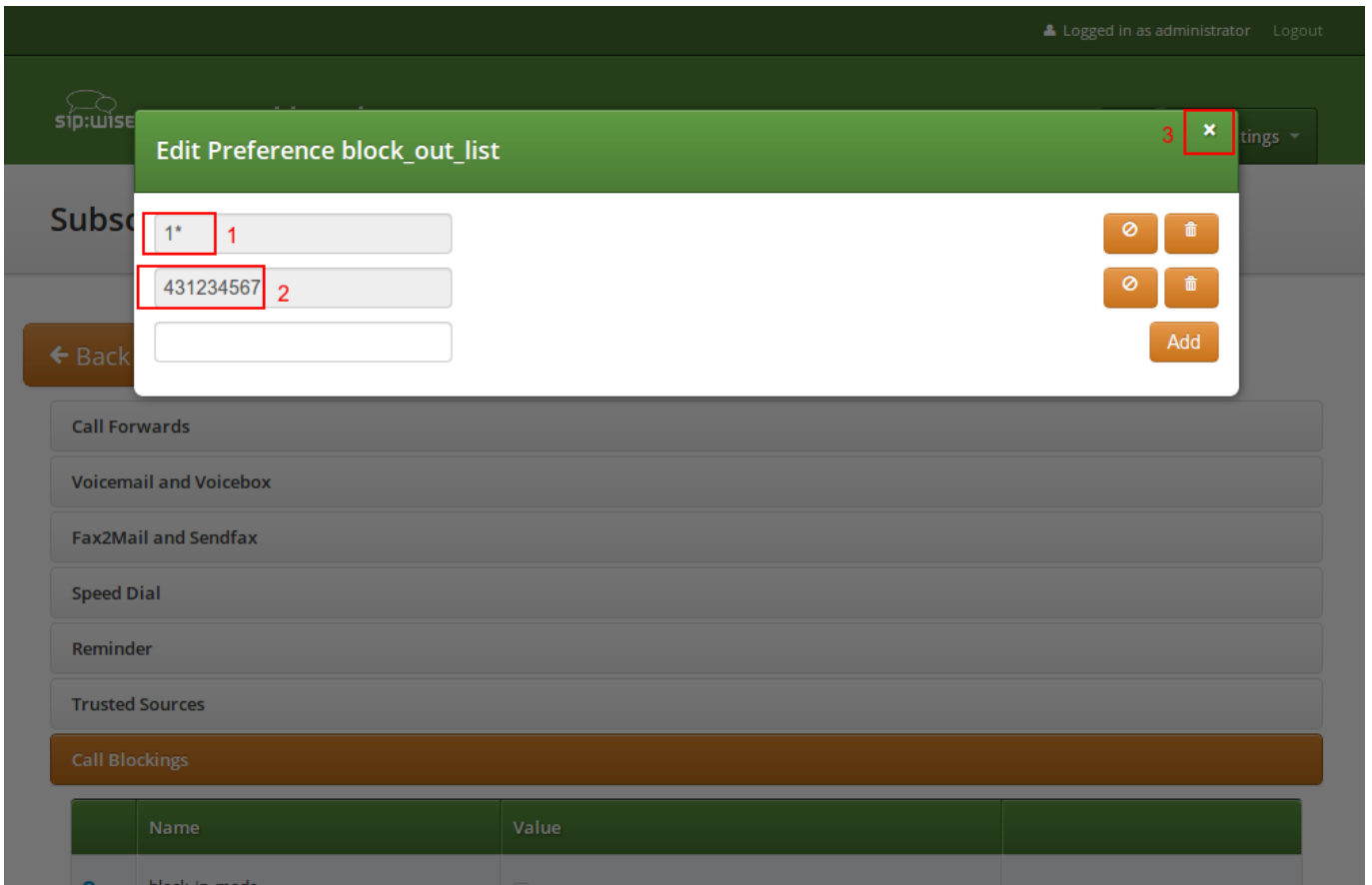
- The *blacklist* mode (option is not checked) tells the system to **allow anything except the entries in the list**. Use this mode if you just want to block certain numbers and allow all the rest.
- The *whitelist* mode indicates to **reject anything except the entries in the list**. Use this mode if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

### 7.3.1.2 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.



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

### 7.3.1.3 Block Anonymous Numbers

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

### 7.3.2 NCOS (Network Class of Service) Levels

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

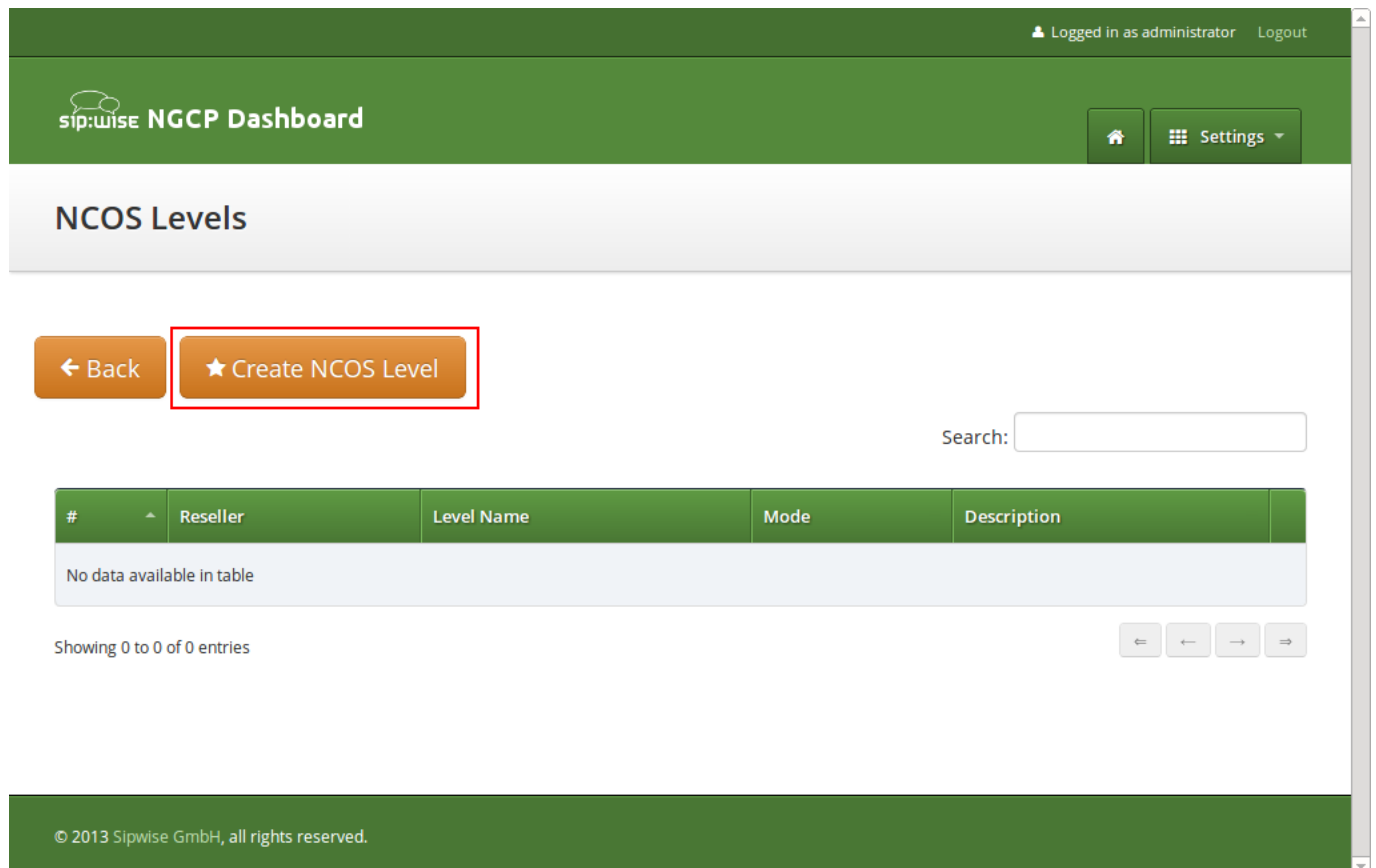
In a case of a conflict, when the Block Lists feature allows a number and NCOS Levels rejects the same number or vice versa, the call will be rejected.

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

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

### 7.3.2.1 Creating NCOS Levels

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



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

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

Reseller

Search:

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

Showing 1 to 1 of 1 entries

Create Reseller

Level Name  2

Mode  3

Description  4

5

© 2013 Sipwise GmbH, all rights reserved.

### 7.3.2.2 Creating Rules per NCOS Level

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

sip:wise NGCP Dashboard

Home Settings

## NCOS Levels

← Back ★ Create NCOS Level

NCOS level successfully created

Search:

| # | Reseller | Level Name | Mode      | Description     |  |
|---|----------|------------|-----------|-----------------|--|
| 1 | default  | test       | blacklist | NCOS Test Level | <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Patterns</a> |

Showing 1 to 1 of 1 entries

© 2013 Sipwise GmbH, all rights reserved.

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

- **NCOS Number Patterns:** here you can define number patterns that will be matched against the called number and allowed or blocked, depending on whitelist / blacklist mode. The patterns are regular expressions.
- **NCOS LNP Carriers:** here you can select predefined *LNP Carriers* that will be allowed (whitelist mode) or prohibited (blacklist mode) to route calls to them. For each of them you can restrict the matching to a predefined number pattern. (See Section 7.6.1 in the handbook for the description of LNP functionality)

---

### Note

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

---

The screenshot shows two sections of a web interface. The top section is titled "NCOS Number Patterns" and contains a "Back" button and a "Create Pattern Entry" button. Below this is a success message: "NCOS pattern successfully created". There is a "Show 5 entries" dropdown and a search box. A table lists one entry with ID 1, Pattern "^439", and Description "Austrian Premium Numbers". Below the table are checkboxes for "Include local area code" and "Intra PBX Calls within same customer", and an "Edit" button. The bottom section is titled "NCOS LNP Carriers" and contains a "Create LNP Entry" button. It also has a "Show 5 entries" dropdown and a search box. A table lists one entry with ID 1, LNP Carrier "LNP\_Carr1", and Description "Rule for LNP Carrier 1".

Figure 27: NCOS Patterns List

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

The screenshot shows a modal window titled "Create Number Pattern". It has a green header with a close button. The form contains two input fields: "Pattern" with the value "^439" and "Description" with the value "Austrian Premium Numbers". A "Save" button is located at the bottom right of the form.

Figure 28: Create NCOS Number Pattern

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

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

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

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

An example of *NCOS LNP Carrier* definition:

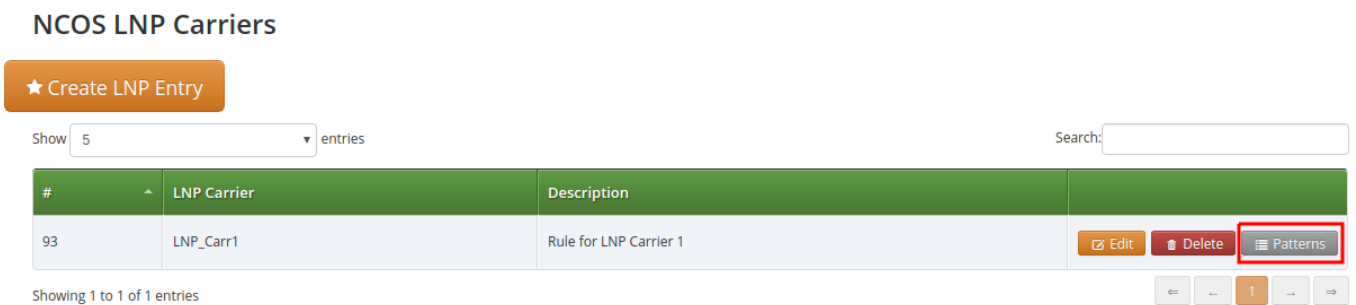


Figure 29: Create NCOS LNP Carrier

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

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

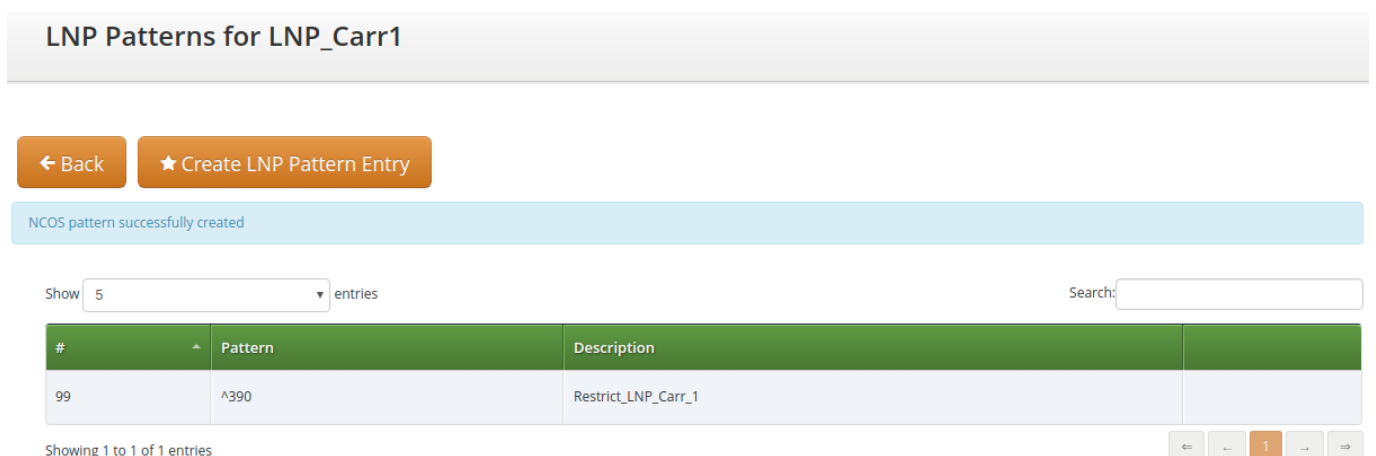


Figure 30: Create NCOS LNP Carrier Pattern

Considering the example before and adding the pattern shown in the picture, the rule now blocks only calls to "LNP\_Carr1" carrier

that starts with 390.

---

### Tip

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

---

The below table gives an overview of all the possible combinations of *NCOS* and *NCOS LNP Carrier*:

Table 3: NCOS combinations

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

- = different behaviour compared with the previous versions (< mr7.5)

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



### 7.3.2.3 Assigning NCOS Levels to Subscribers/Domains

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

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

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

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

### 7.3.2.4 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 Sipwise C5 does not deny provisioning Call Forward to these numbers, the administrator can prevent the incoming calls from being actually forwarded to numbers defined in the NCOS list: just select the appropriate NCOS level in the domain's or subscriber's preference *adm\_cf\_ncos*. This NCOS will apply only to the Call Forward from the subscribers and not to the normal outgoing calls from them.

### 7.3.3 IP Address Restriction

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

By default, *allowed\_ips* is an empty list which means that subscriber is not restricted. If you want to configure a restriction, navigate to *Settings*→*Subscribers*→*Preferences* or *Settings*→*Domains*→*Preferences*, and search for the *allowed\_ips* preference in the *Access Restrictions* section.

| Call Blockings      |                                |                          |                                       |
|---------------------|--------------------------------|--------------------------|---------------------------------------|
| Access Restrictions |                                |                          |                                       |
| 1                   | Name                           | Value                    |                                       |
| ?                   | lock                           |                          |                                       |
| ?                   | concurrent_max                 |                          |                                       |
| ?                   | concurrent_max_out             |                          |                                       |
| ?                   | allowed_clis                   |                          |                                       |
| ?                   | reject_emergency               | <input type="checkbox"/> |                                       |
| ?                   | concurrent_max_per_account     |                          |                                       |
| ?                   | concurrent_max_out_per_account |                          |                                       |
| ?                   | allowed_ips 2                  |                          | 3 <input type="button" value="Edit"/> |
| ?                   | man_allowed_ips                |                          |                                       |
| ?                   | ignore_allowed_ips             | <input type="checkbox"/> |                                       |
| ?                   | allow_out_foreign_domain       | <input type="checkbox"/> |                                       |

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

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

### 7.3.4 CLI-based Access Control

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

The restriction is applied to User-Provided Number (UPN) which is obtained from the configurable source based on the setting of

*inbound\_upn* preference in the *Access Restrictions* section in the Domain and/or User preferences, after it has been rewritten with Inbound Rewrite Rules for Caller.

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

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

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

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

If separate preference *upn\_block\_clir* is enabled, incoming anonymous calls from this user will be dropped.

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

### 7.3.5 Call Limit Control

There's a set of preferences that limits calls to and from subscribers. The option *concurrent\_max\_total* defines the maximum number of concurrent calls (incoming and outgoing) for a subscriber, while the option *concurrent\_max\_out\_total* limits only subscriber's outbound concurrent calls and the option *concurrent\_max\_in\_total* only subscriber's inbound concurrent calls.

Preferences *concurrent\_max*, *concurrent\_max\_out*, and *concurrent\_max\_in* have the same effect, excluding calls to voicemail, application server and intra-PBX calls.

It's also possible to limit the number of concurrent calls of a subscriber compared to the number of calls made or received by all subscribers within the same customer (account). The options *concurrent\_max\_per\_account*, *concurrent\_max\_out\_per\_account*, *concurrent\_max\_in\_per\_account* permit to apply this limit. To better understand how they work, suppose we have two subscribers A and B, owned by the same customer. If we set *concurrent\_max\_per\_account*=2 on B preferences and A is placing two calls, then B can not receive or place new calls at the same time.

When *concurrent\_max\_total* limit is reached, announcement set on *max\_calls\_in* is played to those who try to call that subscriber. The same announcement is played for *concurrent\_max*, *concurrent\_max\_per\_account*, *concurrent\_max\_in\_total*, *concurrent\_max\_in*, *concurrent\_max\_in\_per\_account*. When *concurrent\_max\_out\_total* limit is reached, announcement set on *max\_calls\_out* is played. The same announcement is played for *concurrent\_max\_out* or *concurrent\_max\_out\_per\_account*.

Options *concurrent\_max*, *concurrent\_max\_out* and *concurrent\_max\_in* are configurable on peers as well.

Furthermore, options *concurrent\_max*, *concurrent\_max\_out*, *concurrent\_max\_in* and their *\_total* version (*concurrent\_max\_total* and so on), are configurable on customer as well. In this case the limits are applied considering the number of calls of all the

subscribers belonging to that account.

## 7.4 Call Forwarding and Call Hunting

The Sipwise C5 provides the capabilities for normal *call forwarding* (deflecting a call for a local subscriber to another party immediately or based on events like the called party being busy or doesn't answer the phone for a certain number of seconds) and *serial call hunting* (sequentially executing a group of deflection targets until one of them succeeds). Targets can be stacked, which means if a target is also a local subscriber, it can have another call forward or hunt group which is executed accordingly.

### 7.4.1 Call Forward Types

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

- **Call Forward Unconditional (CFU)**: The call forward is always executed, completely disregarding the subscriber state.
- **Call Forward Busy (CFB)**: The call forward is executed when the subscriber returns a busy state.
- **Call Forward Timeout (CFT)**: The call forward is executed when no answer is received from the subscriber before the timeout expiration. Timeout is configurable in *ringtimeout* subscriber preference.
- **Call Forward Unavailable (CFNA)**: The call forward is executed when the subscriber has no endpoint registered.
- **Call Forward SMS (CFS)**: The SMS forward is always executed, completely disregarding the subscriber state. SMS service has to be enabled, see the [SMS \(Short Message Service\)](#) subchapter for a detailed description on how to activate it.
- **Call Forward on Response (CFR)**: The call forward is executed only for particular reply codes received back from the destination endpoint. The list of the reply codes and the activation mode can be configured in *rerouting\_codes* and *rerouting\_mode* subscriber's preferences. Example: suppose that *rerouting\_codes* is set to 503, *rerouting\_mode* to whitelist and the CFR is configured. If that subscriber receives a call and it replies back a with code 503, then the call will be re-routed to the destination configured in the CFR. For all the other reply codes the CFR will be NOT executed.
- **Call Forward on Overflow (CFO)**: The call forward is executed when the new incoming call for the subscriber exceeds the limit configured in *concurrent\_max\_in\_total*, *concurrent\_max\_in* or *concurrent\_max\_in\_per\_account* subscriber's preferences. If none of the preferences is set then the CFO will be NOT executed.

---

#### Important



Starting from mr7.2.1 release, **Call Forward on Response (CFR)** has to be configured on the **callee** subscriber (in previous versions the preference was associated to the caller subscriber). When the destination endpoint replies back with an error code, this will be matched with the one listed in the *rerouting\_codes* and *rerouting\_mode* callee's preferences.

---

### 7.4.2 Setting a simple Call Forward

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

Edit Call Forward Unconditional
✕

Destination

- Voicemail
- Conference
- Fax2Mail
- Custom Announcement
- Manager Secretary
- URI/Number

URI/Number

for (seconds)

Enabled

Advanced View
Save

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

### 7.4.3 Call Forward Destinations

- **Voicemail:** Calls are forwarded to the Voicemail Application Server where the caller can leave a message.
- **Conference:** Calls are forwarded to the conference room. The subscriber is the host of the conference.
- **Fax2Mail:** Calls are forwarded to the Fax Server and the caller is supposed to leave a fax message. Note: The Fax2Mail feature must be enabled in the subscriber's preferences.
- **Custom Announcement:** A custom announcement is played back to the caller. Select an announcement from the *Custom announcement* list.
- **Manager Secretary:** Calls are forwarded to numbers defined in the "manager\_secretary\_numbers" subscriber preference. The "manager\_secretary" feature must be enabled.
- **URI/Number:** The call is forwarded to the provided SIP-URI string or a number (See the *Call Forward Destination Extra Parameters* section below).

#### 7.4.3.1 Call Forward Destination Options

- **URI/Number:** A destination to forward calls to. This option is only valid for the *URI/Number* destination type. Specify a valid SIP-URI string or a plain number.

- **for (seconds)**: Sets the ringing time, after which the call is forwarded to the next number on the list (if configured).
- **Custom Announcement**: Custom Announcements are created in Sound Sets and must have the name like *custom\_announcement\_0*, where the trailing symbol is a digit from 0 to 9.
- **Enabled**: Defines whether the Call Forward rule is being used or not.

#### 7.4.4 Advanced Call Hunting

Beside call forwarding to a single destination, Sipwise C5 offers the possibility to activate call forwarding in a more sophisticated way:

- to multiple destinations (→ *Destination Set*)
- only during a pre-defined time set (→ *Time Set*)
- only for specific callers (→ *Source Set*)
- only for specific callee (→ *B-Number Set*)

If you want to define such more detailed call forwarding rules, you need to change into the *Advanced View* when editing your call forward. There, you can select multiple *Destination Set - Time Set - Source Set - B-Number Set* groups that determine all conditions under which the call will be forwarded.

##### Explanation of call forward parameters

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

##### 7.4.4.1 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.

### Edit Destination Set ✕

Name

Destination  Voicemail  
 Conference  
 Custom Announcement  
 URI/Number

URI/Number

for (seconds)

Priority

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

### Edit Call Forward Unconditional ✕

during Time Set

from Source Set

to B-Number Set

Destination Set

Enabled

Press *Save* to store your settings.

#### 7.4.4.2 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.



**Edit Time Set** ✕

Name

| Period | Year    | Month      | Day     | Weekday  | Hour    | Minute  |
|--------|---------|------------|---------|----------|---------|---------|
|        | 2014 ▾  | April ▾    | ▾       | Monday ▾ | ▾       | ▾       |
|        | through | through    | through | through  | through | through |
|        | ▾       | Septembe ▾ | ▾       | Friday ▾ | ▾       | ▾       |

Remove

Add another period

Save

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

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



#### Important

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

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

#### 7.4.4.3 Configuring Source Sets

Once the *Advanced View* of the call forward definition has been opened, you will need to press the *Manage Source Sets* button to start defining new Source Sets or managing an existing one. The following image shows the Source Set definition dialog:

The screenshot shows the 'Edit Source Set' interface. At the top is a green header with a close button (X). Below the header, there are several input fields:
 

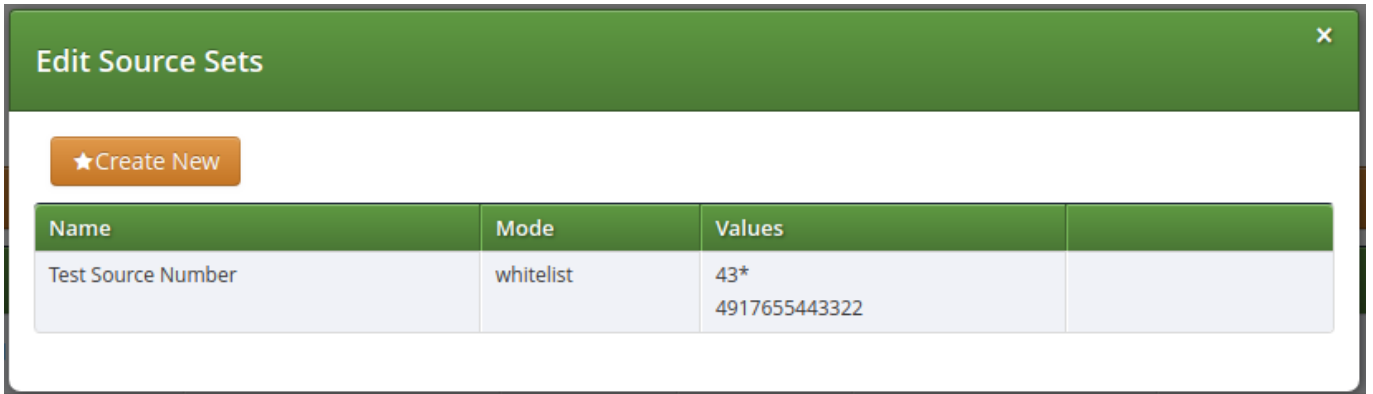
- Name:** A text input field containing 'Test Source Number'.
- Mode:** A dropdown menu currently set to 'whitelist'.
- Is regex:** A checkbox that is currently unchecked.
- Source:** Two text input fields. The first contains '43\*' and the second contains '4917655443322'. Both fields are highlighted with red boxes. To the right of each Source field is an orange 'Remove' button.
- Buttons:** An orange 'Add another source' button is located below the second Source field. At the bottom right of the form is an orange 'Save' button.

Figure 31: Creating a Call Forward Source Set

You will need to fill in the `Name` field first, the `Mode: whitelist` or `blacklist`, the `is_regex` flag and finally in the `Source` field you can enter:

- A simple phone number in E.164 format
- A pattern, in order to define a range of numbers. You can use "\*" (matches a string of 0 to any number of characters), "?" (matches any single character), "[abc]" (matches a single character that is part of the explicitly listed set: a, b or c) and "[0-9]" (matches a single character that falls in the range 0 to 9) as wildcards, as usual in shell patterns. Examples:
  - "431\*" (all numbers from Vienna / Austria)
  - "49176[0-5]77\*" (German numbers containing fixed digits and a variable digit in 0-5 range in position 6)
  - "43130120??" (numbers from Vienna with fixed prefix and 2 digits variable at the end)
- A perl compatible regular expressions (only if `is_regex` if set). Capturing groups can be formed using parentheses and referenced in the *Destination Set* via `\1`, `\2`,...
- The constant string "anonymous" that indicates a suppressed calling number (CLIR)

You can add more patterns to the Source Set by pressing the *Add another source* button. When you finished adding all patterns, press the *Save* button. You will then see the below depicted list of Source Sets:

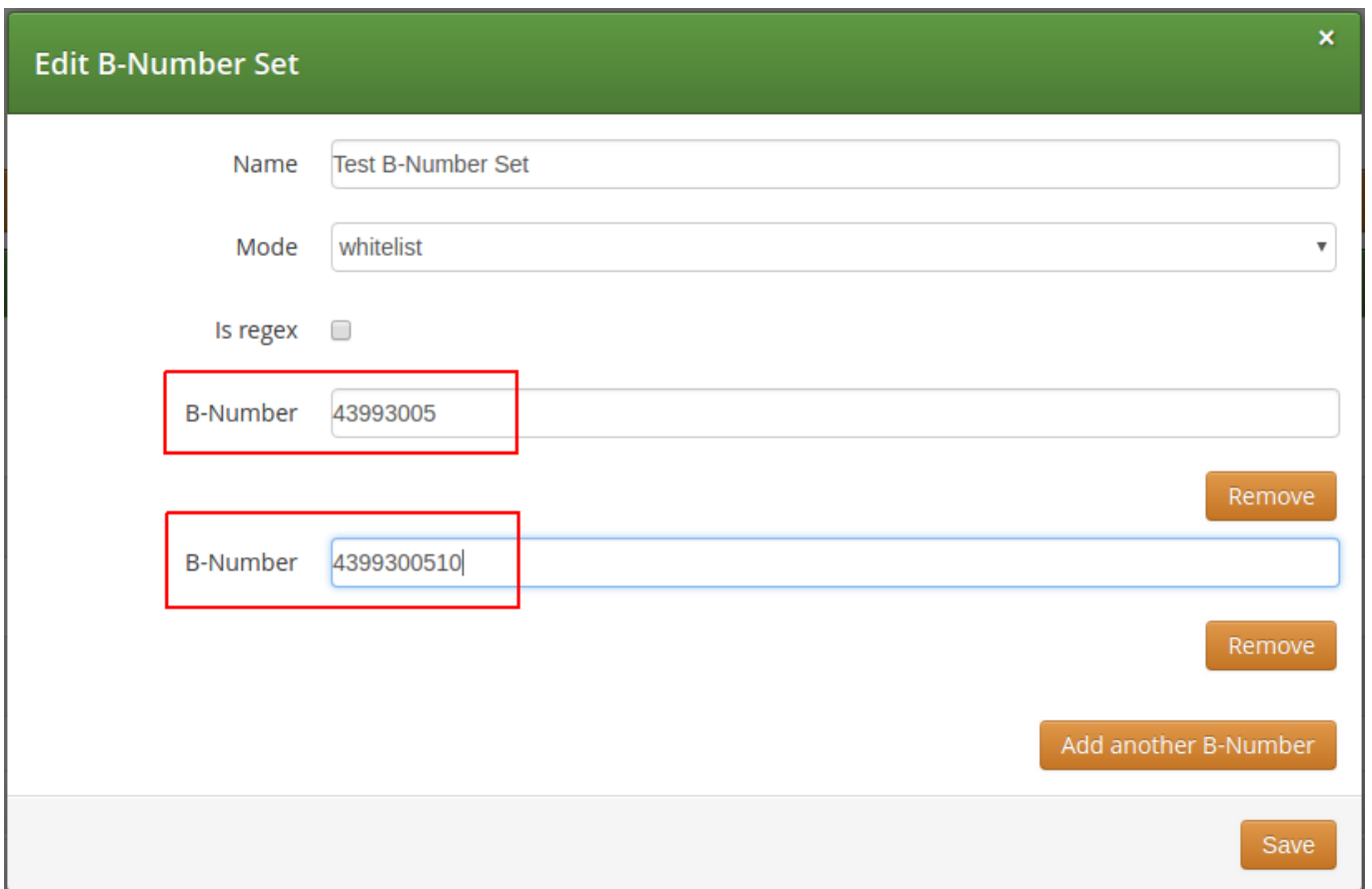


| Name               | Mode      | Values               |
|--------------------|-----------|----------------------|
| Test Source Number | whitelist | 43*<br>4917655443322 |

Figure 32: List of Call Forward Source Sets

#### 7.4.4.4 Configuring B-Number Sets

Once the *Advanced View* of the call forward definition has been opened, you will need to press the *Manage B-Number Sets* button to start defining new B-Number Sets or managing an existing one. The following image shows the B-Number Set definition dialog:



Name: Test B-Number Set

Mode: whitelist

Is regex:

B-Number: 43993005

B-Number: 4399300510

Remove

Remove

Add another B-Number

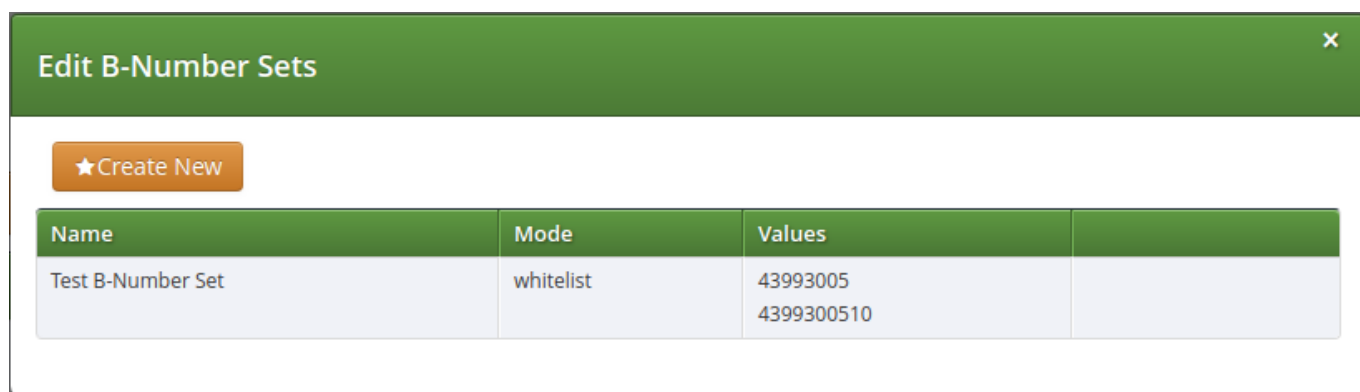
Save

Figure 33: Creating a Call Forward B-Number Set

You will need to fill in the `Name` field first, the `Mode: whitelist` or `blacklist`, the `is_regex` flag and finally in the `B-Number` field you can enter:

- A simple phone number in E.164 format
- A pattern, in order to define a range of numbers. You can use "\*" (matches a string of 0 to any number of characters), "?" (matches any single character), "[abc]" (matches a single character that is part of the explicitly listed set: a, b or c) and "[0-9]" (matches a single character that falls in the range 0 to 9) as wildcards, as usual in shell patterns. Examples:
  - "431\*" (all numbers from Vienna / Austria)
  - "49176[0-5]77\*" (German numbers containing fixed digits and a variable digit in 0-5 range in position 6)
  - "43130120??" (numbers from Vienna with fixed prefix and 2 digits variable at the end)
- A perl compatible regular expressions (only if `is_regex` if set). Capturing groups can be formed using parentheses and referenced in the *Destination Set* via `\1`, `\2`,...

You can add more patterns to the B-Number Set by pressing the *Add another B-Number* button. When you finished adding all patterns, press the *Save* button. You will then see the below depicted list of B-Number Sets:



| Edit B-Number Sets |           |                        |  |
|--------------------|-----------|------------------------|--|
| Name               | Mode      | Values                 |  |
| Test B-Number Set  | whitelist | 43993005<br>4399300510 |  |

Figure 34: List of Call Forward B-Number Sets

#### 7.4.4.5 Finalizing the call forward definition

As additional step you can define a Destination Set as described in [Destination Sets](#) subchapter. For our example, we have defined the following Destination Set:

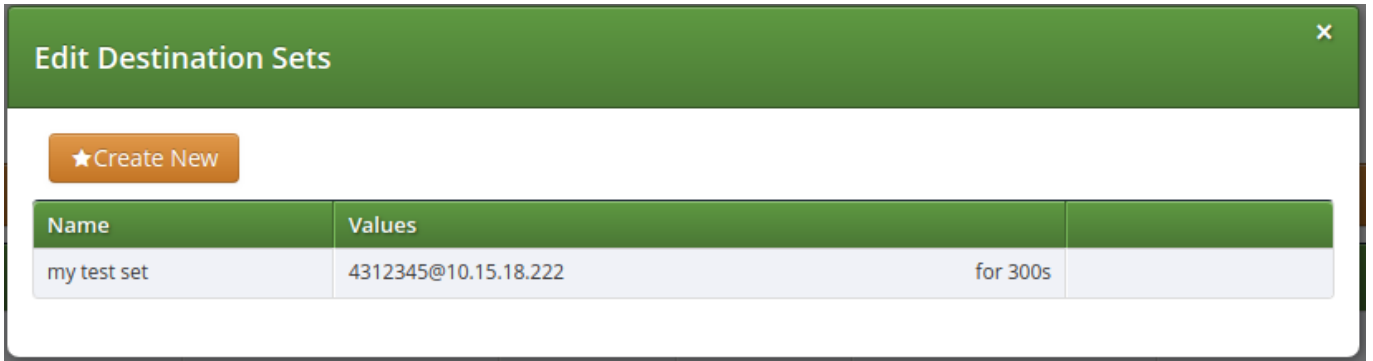


Figure 35: List of Call Forward Destination Sets

A final step of defining the call forward settings is selecting a Destination, a Time Set, a Source Set and a B-Number Set, as shown in the image below. *Please note* that there is no specific Time Set selected in our example, that means the call forward rule is valid (as shown) <always>.

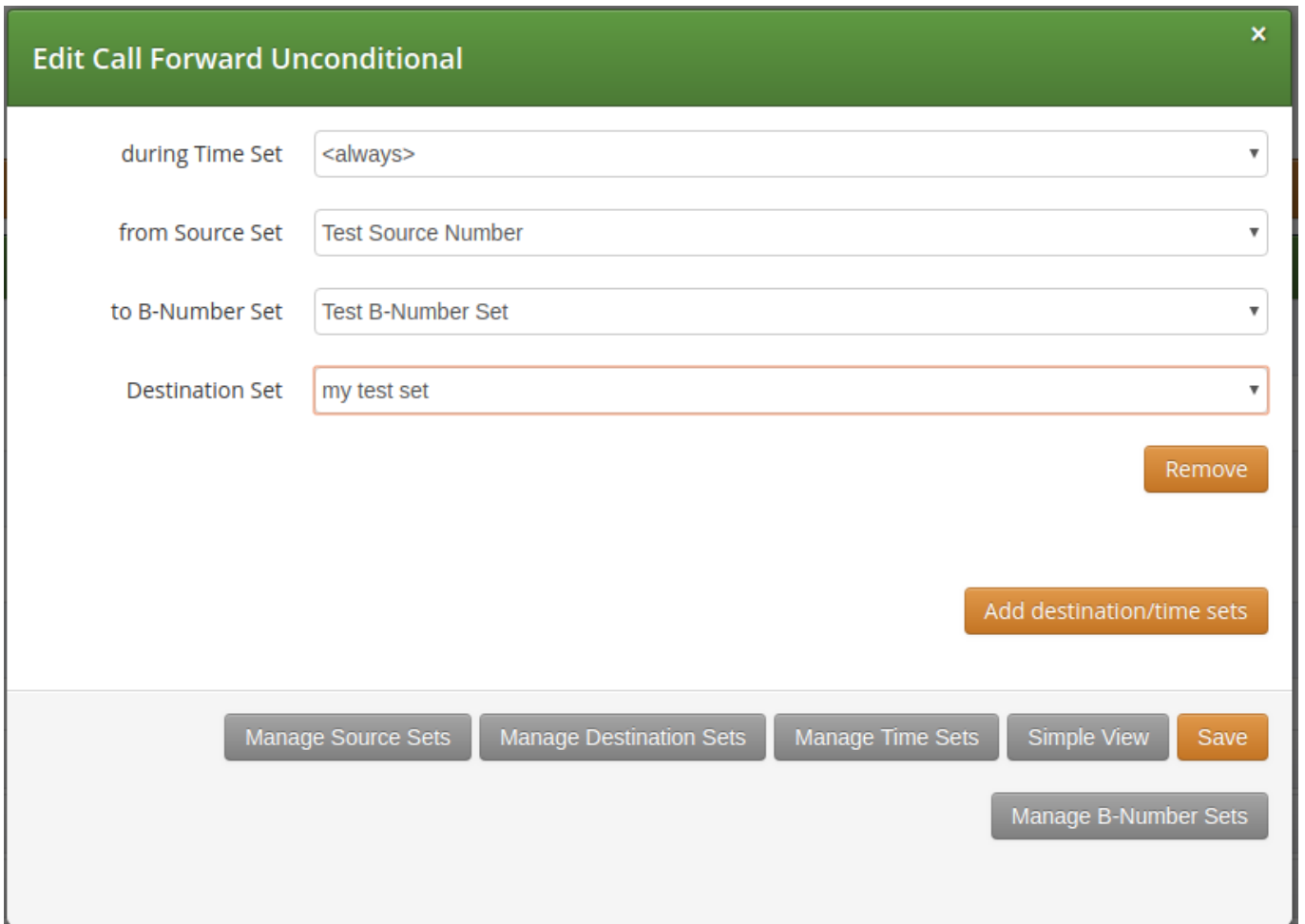
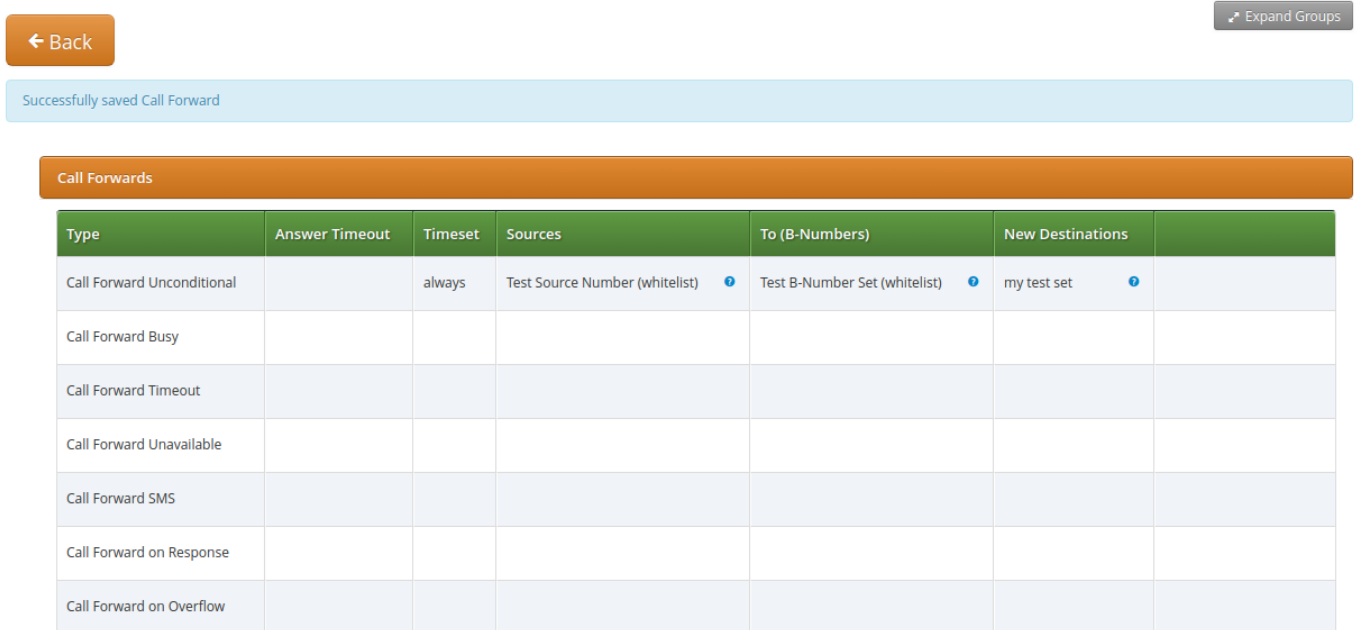


Figure 36: Definition of a Call Forward with Source and Destination Sets

Once all the settings have been defined and the changes are saved, you will see the call forward entry (in our example: *Call Forward Unconditional*), with the names of the selected Destination, Time Set, Source Sets and B-Number Set provided, at *SubscriberPreferences* → *Call Forwards* location on the web interface:



← Back Expand Groups

Successfully saved Call Forward

| Type                       | Answer Timeout | Timeset | Sources                        | To (B-Numbers)                | New Destinations |
|----------------------------|----------------|---------|--------------------------------|-------------------------------|------------------|
| Call Forward Unconditional |                | always  | Test Source Number (whitelist) | Test B-Number Set (whitelist) | my test set      |
| Call Forward Busy          |                |         |                                |                               |                  |
| Call Forward Timeout       |                |         |                                |                               |                  |
| Call Forward Unavailable   |                |         |                                |                               |                  |
| Call Forward SMS           |                |         |                                |                               |                  |
| Call Forward on Response   |                |         |                                |                               |                  |
| Call Forward on Overflow   |                |         |                                |                               |                  |

Figure 37: List of Call Forward with Source and Destination Sets

## 7.5 Call Forking by Q value

The Sipwise C5 platform allows you to register multiple devices under the same subscriber. By the default, the maximum number of the device you can register is 5. This value is configurable via `kamailio→proxy→max_registrations_per_subscriber` preference in `config.yml`.

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

### 7.5.1 The Q value

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

In case the client doesn't set the q value, Sipwise C5 just set a default value of `q=-1` in the database.

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

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

### 7.5.2 The Standard Method

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

---

#### Note

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

---

This method can be activated setting *Standard* in subscriber/domain preference *serial\_forking\_by\_q\_value*.

### 7.5.3 The Probability Method

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

---

#### Note

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

---

This method can be activated setting *Probability* in subscriber/domain preference *serial\_forking\_by\_q\_value*.

### 7.5.4 Advanced Configurations

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

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

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

---

**Note**

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

---

## 7.6 Local Number Porting

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

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

---

**Note**

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

---

### 7.6.1 Local LNP Database

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

#### 7.6.1.1 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 Sipwise C5. This is useful to define your own number blocks, and in case of calls to those numbers reject the calls if the numbers are not assigned to local subscribers (otherwise they would be routed to a peer, which might cause call loops). Finally the *Skip Rewrite* flag skips executing of LNP Rewrite Rules if no number manipulation is desired for an LNP carrier.

#### 7.6.1.2 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. For external billing purposes, the optional *Type* tag of a matched LNP number is recorded in CDRs.

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

### 7.6.1.3 Enabling local LNP support

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

### 7.6.1.4 LNP Routing Procedure

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

If the called party number is not assigned to a local subscriber, or if the called party is a local subscriber and has the subscriber/-domain preference *lnp\_for\_local\_sub* set, the LNP lookup logic is engaged, otherwise the call proceeds without LNP lookup. The further steps assume that LNP is engaged.

If the call originated from a peer, and the peer preference *caller\_lnp\_lookup* is set for this peer, then an LNP lookup is performed using the normalized calling party number. The purpose for that is to find the LNP prefix of the calling peer, which is then stored as *source\_lnp\_prefix* in the CDR, together with the selected LNP number's *type* tag (*source\_lnp\_type*). If the LNP lookup does not return a result (e.g. the calling party number is not populated in the local LNP database), but the peer preference *default\_lnp\_prefix* is set for the originating peer, then the value of this preference is stored in *source\_lnp\_prefix* of the CDR.

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

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

**Important**

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

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

- `${callee_lnp_prefix}`: The prefix stored in the LNP Carrier
- `${callee_lnp_basenum}`: The actual number entry causing the match (may be shorter than the called party number due to longest match)

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

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

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

**7.6.1.5 Blocking Calls Using LNP Data**

The Sipwise C5 provides means to allow or block calls towards ported numbers that are hosted by particular LNP carriers. Please visit Section [7.3.2.2](#) in the handbook to learn how this can be achieved.

**7.6.1.6 Transit Calls using LNP**

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

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

### 7.6.1.7 CSV Format

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

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

Table 4: LNP CSV Format

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

### 7.6.1.8 Local LNP returned values

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

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

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

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

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

## 7.7 Emergency Mapping

As opposed to the [Simple Emergency Number Handling](#) solution, Sipwise C5 supports an advanced emergency call handling method, called *emergency mapping*. The main idea is: instead of obtaining a statically assigned emergency prefix / suffix from subscriber preferences, Sipwise C5 retrieves an emergency routing prefix from a central emergency call routing table, according to the current location of the calling subscriber.

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

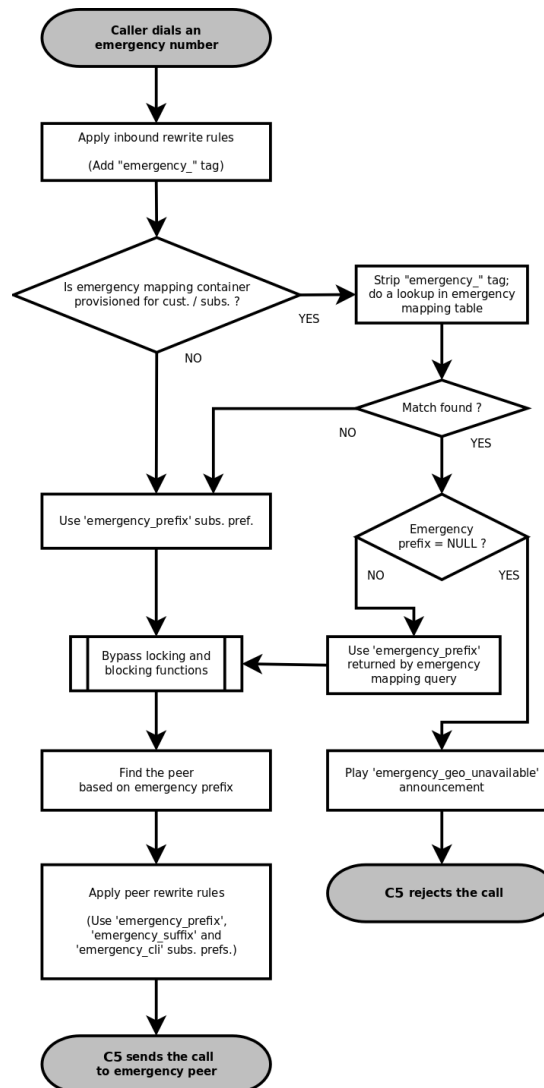


Figure 38: Emergency Call Handling with Mapping

### 7.7.1 Emergency Mapping Description

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

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

The actual emergency number mapping is done per location (per *Emergency Container*), using the so-called *Emergency Mapping* entries. An *Emergency Mapping* entry assigns a routing prefix, valid only in a 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, Sipwise C5 performs an exact match on the emergency number in the emergency routing table.

---

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

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

---

**Note**

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

---

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

### 7.7.2 Emergency Mapping Configuration

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

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

✕

## Edit Emergency Containers

Reseller Search:

| # | Name                       | Contract # | Status |                                     |
|---|----------------------------|------------|--------|-------------------------------------|
| 1 | default                    | 1          | active | <input checked="" type="checkbox"/> |
| 3 | api_test test reseller     | 137        | active | <input type="checkbox"/>            |
| 5 | patched name 1494894408    | 179        | active | <input type="checkbox"/>            |
| 7 | test reseller 1494894408 2 | 181        | active | <input type="checkbox"/>            |
| 9 | test reseller 1494894408 3 | 183        | active | <input type="checkbox"/>            |

Showing 1 to 4 of 8 entries

Name

Figure 39: Creating an Emergency Container

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

---

### Tip

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

---

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

Emergency Mapping Container

Search:

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

Showing 1 to 2 of 2 entries

Create Emergency Mapping Container

Code

Prefix

Save

Figure 40: Creating an Emergency Mapping Entry

The following parameters must be set:

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

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

## Emergency Mappings

← Back
★ Download CSV
★ Upload CSV

### Emergency Containers

★ Create Emergency Container

Show 5 entries Search:

| # | Reseller | Name        |
|---|----------|-------------|
| 1 | default  | EmergCont_1 |
| 3 | default  | EmergCont_2 |

Showing 1 to 2 of 2 entries

### Emergency Mappings

★ Create Emergency Mapping

Show 5 entries Search:

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

Figure 41: Emergency Mapping List

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



Customer #205 - Preferences

← Back Expand Groups

Call Blockings

Access Restrictions

**Number Manipulations**

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

Internals

Figure 42: Assigning an Emergency Mapping Container

### Rewrite Rules for Emergency Mapping

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



#### Important

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

#### 7.7.2.1 Emergency Calls Not Allowed

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

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

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

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

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

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

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

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

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

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

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

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

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

Upload mapping  (None)

Reseller

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

Showing 1 to 4 of 8 entries

Purge existing

Figure 43: Uploading Emergency Mapping Data

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

## 7.8 Emergency Priorization

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

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

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

- Any SIP requests (calls, registrations etc.) from subscribers within the affected domains, who are not marked as privileged, are rejected.
- Any calls from peers not targeting privileged subscribers are rejected.

- Any active calls which do not have a privileged subscriber involved are terminated.

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

### 7.8.1 Call-Flow with Emergency Mode Enabled

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

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

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

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

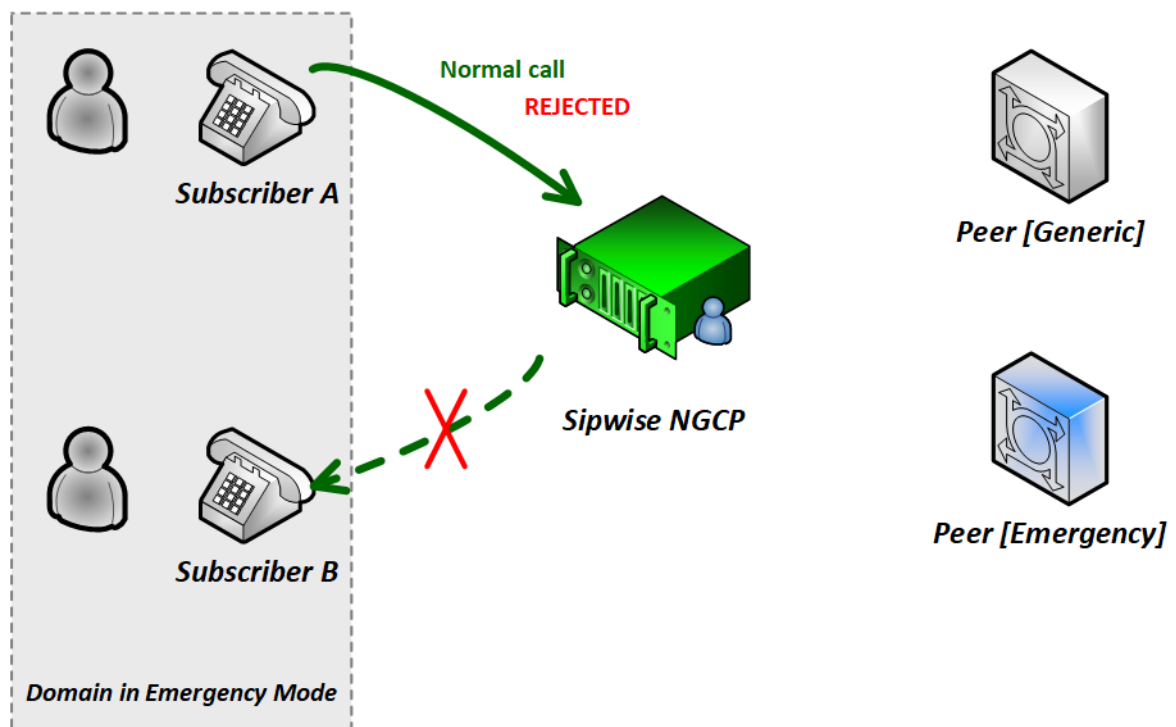


Figure 44: Call-flow in Emergency Mode 1. (Std to Std)

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

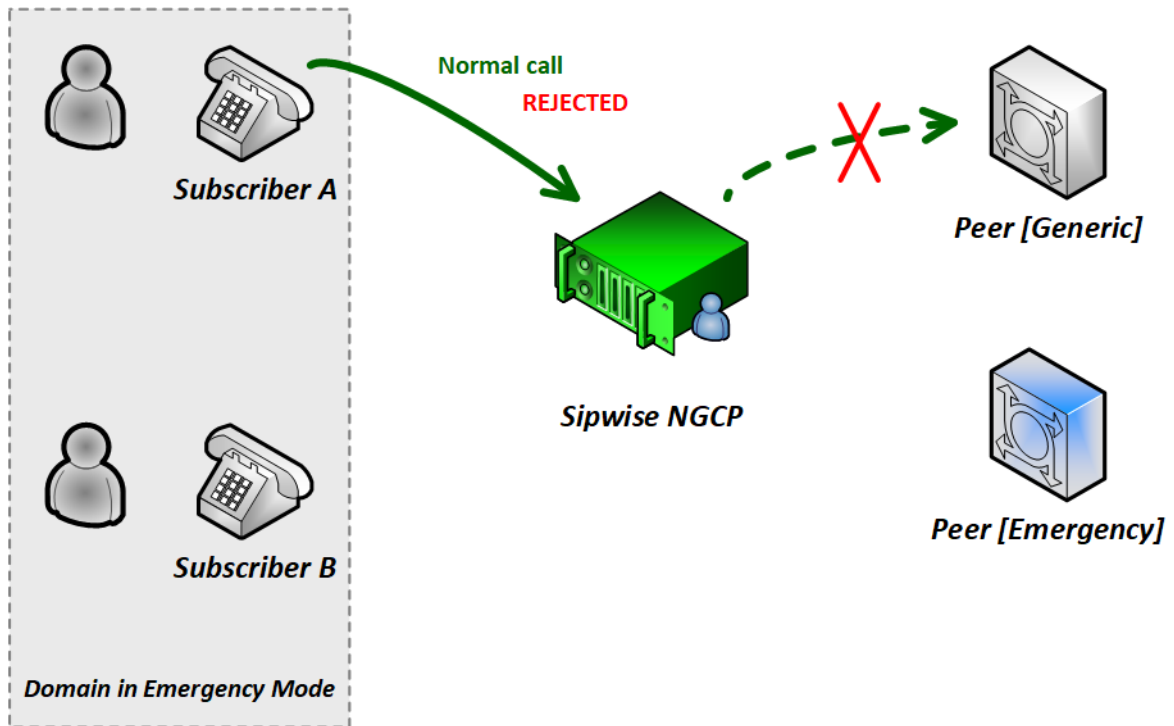


Figure 45: Call-flow in Emergency Mode 2. (Std to Peer)

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

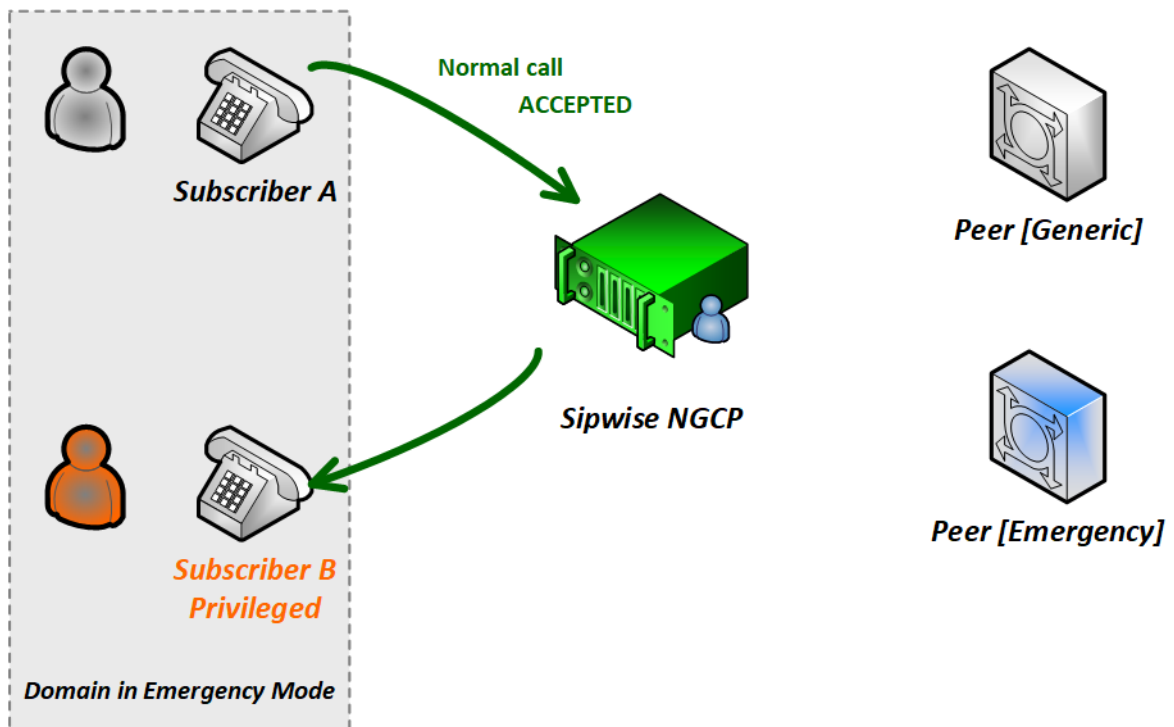


Figure 46: Call-flow in Emergency Mode 3. (Std to Priv)

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

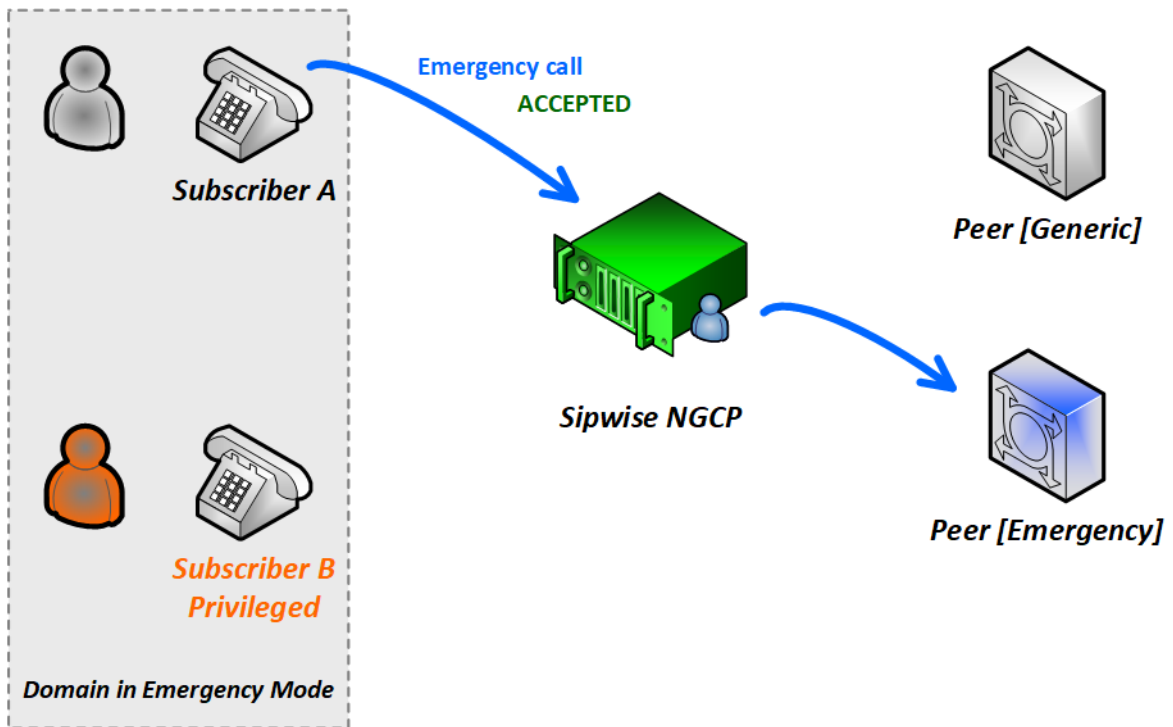


Figure 47: Call-flow in Emergency Mode 4. (Std to Emerg)

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

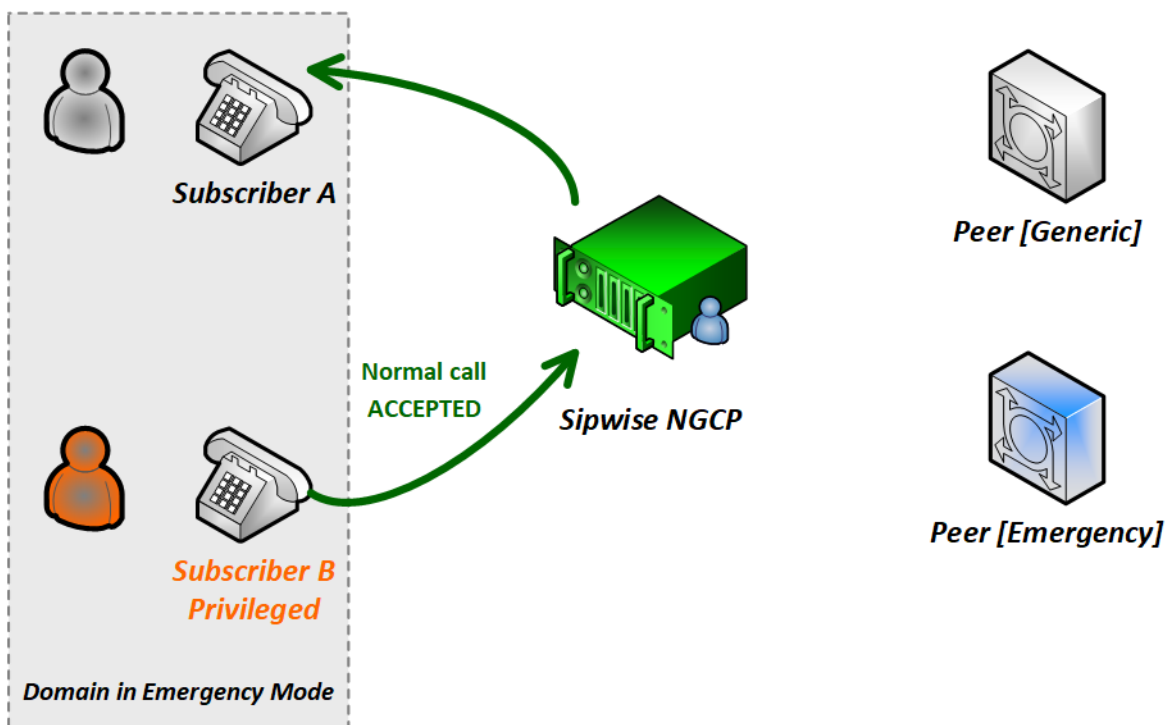


Figure 48: Call-flow in Emergency Mode 5. (Priv to Std)

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

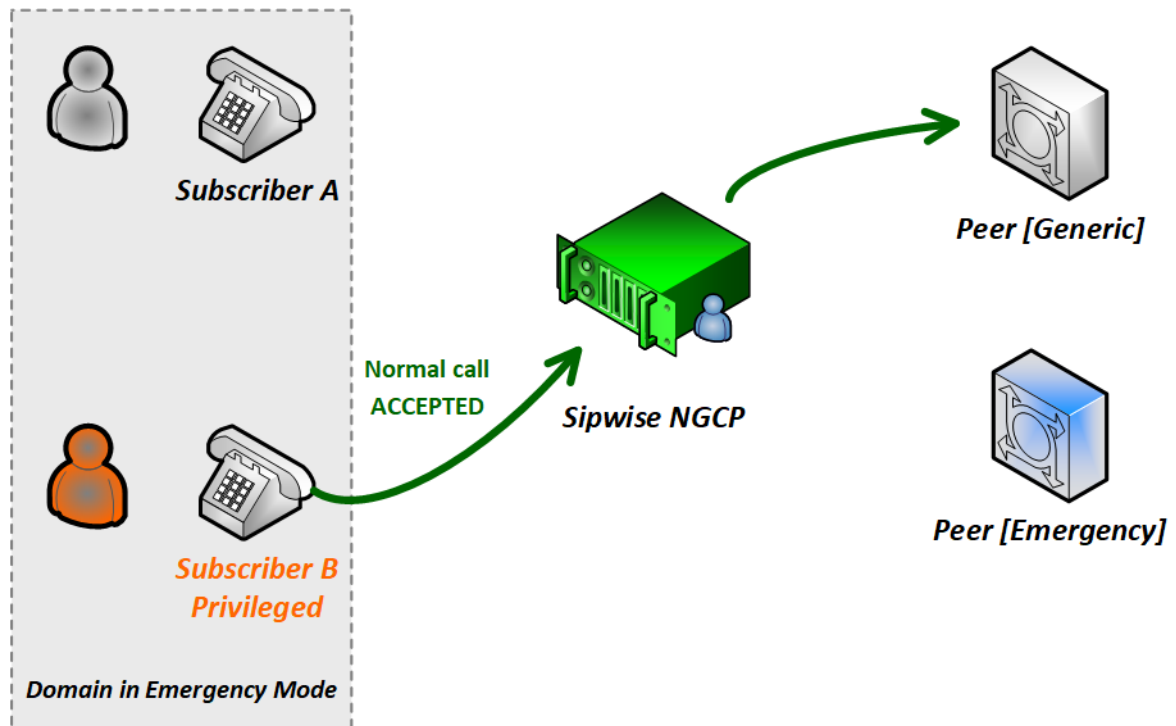


Figure 49: Call-flow in Emergency Mode 6. (Priv To Peer)

## 7.8.2 Configuration of Emergency Mode

The platform operator has to perform 2 steps of configuration so that the emergency mode can be activated. After the configuration is completed it is necessary to explicitly activate emergency mode, which can be accomplished as described in Section 7.8.3 later.

### 1. System-level Configuration

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

In order to learn about other parameters related to emergency prioritization please refer to Section B.1.13 part of the handbook.

### 2. Subscriber-level Configuration

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

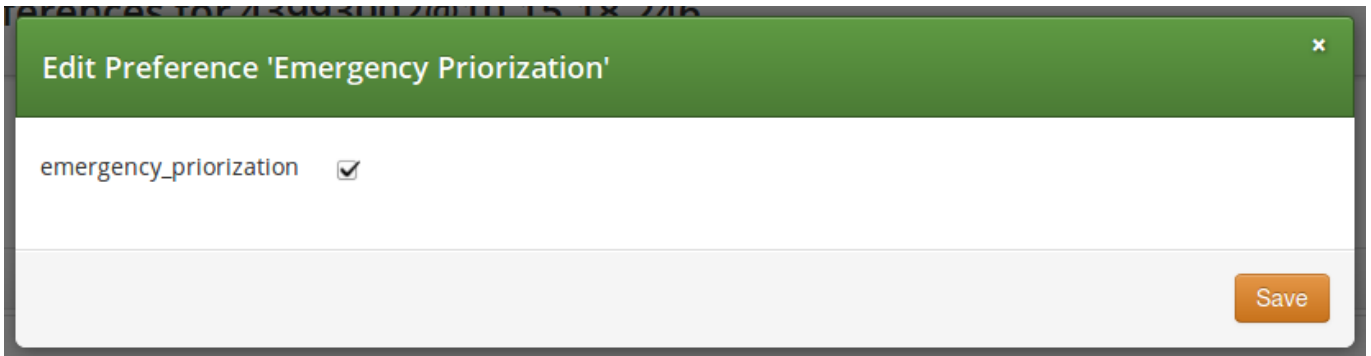


Figure 50: Emergency Priorization of Subscriber

The checkbox `emergency_priorization` has to be ticked and then press the `Save` button.

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

### 7.8.3 Activating Emergency Mode

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

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



#### Important

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

---

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



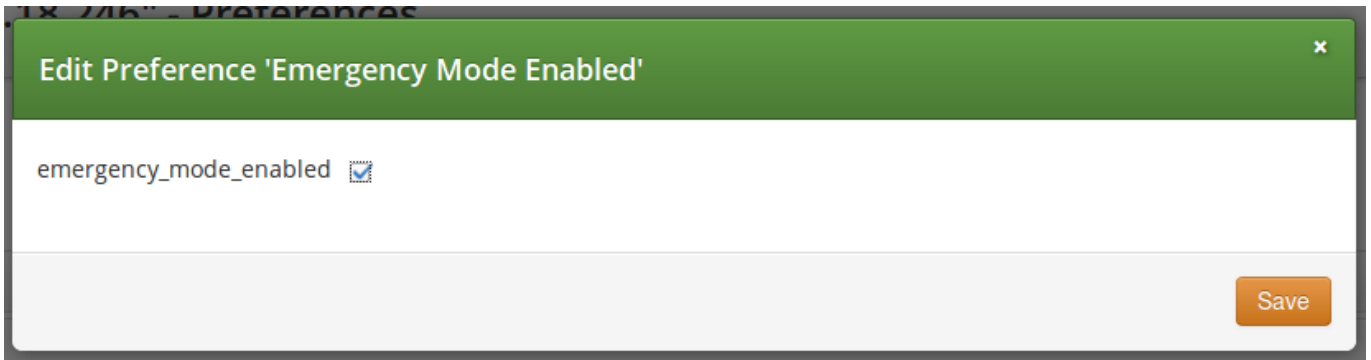


Figure 51: Activate Emergency Mode of Domain

The checkbox `emergency_mode_enabled` has to be ticked and then press the `Save` button.

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

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

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

- Enable emergency mode:

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

- Disable emergency mode:

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

- Query the status of emergency mode:

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

## 7.9 SIP Message Filtering

### 7.9.1 Header Filtering

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

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

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



### Warning

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

## 7.9.2 Codec Filtering

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

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

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

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

In this way the G729 and G722 will be removed from the SDP payload. In order to apply the changes, run

```
ngcpcfg apply 'Enable CODEC filtering'
```

### 7.9.3 Enable History and Diversion Headers

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

- Domain's Preferences: `inbound_uprn` = **Forwarder's NPN**
- Peer's Preferences: `outbound_history_info` = **UPRN**
- Peer's Preferences: `outbound_diversion` = **UPRN**
- Domain's Preferences: `outbound_history_info` = **UPRN** (if you want to allow History Header for on-net call as well)
- Domain's Preferences: `outbound_diversion` = **UPRN** (if you want to allow Diversion Header for on-net call as well)

### 7.9.4 User Agent Filtering

It could be useful to filter the received REGISTER and INVITE messages based on the User Agent header, for example if you want to force your subscribers to use certain types of devices. To achieve that configuration system wide you just need to change the `/etc/ngcp-config/config.yml`, in the following section:

```
kamailio:
  proxy:
    block_useragents:
      action: reject
      enable: yes
      mode: whitelist
      ua_patterns:
        - Yealink.*
```

In the example above, the system is allowing all the messages which have User Agent header starting with *Yealink*. All the others will be rejected with a *403 Forbidden message*. To silently drop the received message it is possible to specify the *drop* action instead of the default *reject*. Another example is the `blacklist` mode:

```
kamailio:
  proxy:
    block_useragents:
      action: drop
      enable: yes
      mode: blacklist
      ua_patterns:
        - friendly-scanner
```

In this example the system will block all the messages which have User Agent header equal to *friendly-scanner*. Because of the *drop* action this messages will be silently dropped, without providing any feedback to the sender. As usual, in order to apply the changes, run

```
ngcpcfg apply 'Enable User-Agent filtering'
```

Regardless of the system-wide configuration (UA filtering enabled or not), it is possible to define a specific User Agent filtering for each Domain or Subscriber. In order to do so, you should configure the following fields in Domain's or Subscriber's Preferences:

- `ua_filter_list`: Contains wildcard list of allowed or denied SIP User-Agents matched against the User-Agent header.
- `ua_filter_mode`: Specifies the operational mode of the SIP User-Agent Filter List: Blacklist or Whitelist.
- `ua_reject_missing`: Rejects any request if no User-Agent header is given.

In case of rejection a message with code `kamailio.proxy.early_rejects.block_admin.announce_code` and reason `kamailio.proxy.early_rejects.block_admin.announce_reason` will be sent back to the subscriber.

## 7.10 SIP Trunking with SIPconnect

### 7.10.1 User provisioning

For the purpose of external SIP-PBX interconnect with Sipwise C5 the platform admin should create a subscriber with multiple aliases representing the numbers and number ranges served by the SIP-PBX.

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

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

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

### 7.10.2 Inbound calls routing

Enable preference *Number Manipulations*→*e164\_to\_ruri* for routing inbound calls to SIP-PBX. This ensures that the Request-URI will comprise a SIP-URI containing the dialed alias-number as user-part, instead of the user-part of the registered AOR (which is normally a static value).

### 7.10.3 Number manipulations

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

### 7.10.3.1 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 6.7. While the SIP-PBX trunk configuration can be sometimes amended it is a good idea in sense of SIPconnect recommendation to send only the global E.164 numbers.

Moreover, in mixed environments with Sipwise C5 Cloud PBX sharing the same domain with SIP trunking (SIP-PBX) customers the subscribers may have different rewrite rules sets assigned to them. The difference is caused by the fact that the dialplan for Cloud PBX is fundamentally different from the dialplan for SIP trunks due to extension dialing, where the Cloud PBX subscribers use the break-out code 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.

#### 7.10.3.2 User parameter

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

- Domain's Preferences: `outbound_from_user_is_phone = Y`
- Domain's Preferences: `outbound_to_user_is_phone = Y`

#### 7.10.3.3 Forwarding number

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

- Domain's Preferences: `inbound_uprn = Forwarder's NPN`
- Domain's Preferences: `outbound_from_user = UPRN (if set) or User-Provided Number`
- Domain's Preferences: `outbound_pai_user = UPRN (if set) or Network-Provided Number`
- Domain's Preferences: `outbound_history_info = UPRN` (if the called user expects History-Info header)
- Domain's Preferences: `outbound_diversion = UPRN` (if the called user expects Diversion header)
- Domain's Preferences: `outbound_to_user = Original (Forwarding) called user` if the callee expects the number of the subscriber forwarding the call, otherwise leave default.

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

---

**Tip**

On outgoing call from SIP-PBX subscriber the Network-Provided Number (NPN) is set to the *cli* preference prefilled with main E.164 number. In order to have the full alias number as NPN on outgoing call set preference *extension\_in\_npn* = Y.

---

**Externally forwarded call** If the call forward takes place inside the SIP-PBX it can use one of the following specification for signaling the diversion number to the platform:

- using **Diversion** method (RFC 5806): configure Subscriber's Preferences: *inbound\_uprn* = **Forwarder's NPN / Received Diversion**
- using **History-Info** method (RFC 7044): Sipwise C5 platform extends the History-Info header received from the PBX by adding another level of indexing according to the specification RFC 7044.

**7.10.3.4 Allowed CLIs**

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

**7.10.4 Registration**

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

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

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

**Important**

The use of RFC 6140 style "bulk number registration" is discouraged. The SIP-PBX should register one AOR with email-style SIP URI. The Sipwise C5 will take care of routing the aliases to the AOR with *e164\_to\_ruri* preference.

---

**7.10.4.1 Trusted Sources**

If a SIP-PBX cannot perform the digest authentication, you can authenticate it by its source IP address in Sipwise C5. To configure the IP-based authentication, go to the subscriber's preferences (*Details*→*Preferences*→*Trusted Sources*) and specify the IP address of the SIP-PBX in the *Source IP* field.

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

When this feature is configured for a subscriber, Sipwise C5 authenticates all calls that arrive from the specified IP address without challenging them.

**Important**

If the same IP address and the FROM field are mistakenly specified as trusted for different subscribers, Sipwise C5 will not know which subscriber to charge for the call and will randomly select one.

---

## 7.11 Trusted Subscribers

In some cases, when you have a device that cannot authenticate itself against Sipwise C5, you may need to create a *Trusted Subscriber*. Trusted Subscribers use IP-based authentication and they have a Permanent SIP Registration URI in order to receive messages from Sipwise C5.

In order to make a regular subscriber trusted, perform the following extra steps:

- Create a permanent registration via (*Subscribers*→*Details*→ *Registered Devices*→*Create Permanent Registration*)
- Add the IP address of the device as Trusted Source in your subscriber's preferences (*Details*→*Preferences*→*Trusted Sources*).

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

## 7.12 Peer Probing

The basic way of selecting the appropriate peering server, where an outbound call can be routed to, has already been described in Section 6.6.2.3 of the handbook.

This chapter provides information on the *peer probing* feature of Sipwise C5 that is available since the mr5.4.1 release.

### 7.12.1 Introduction to Peer Probing Feature

The Sipwise C5 provides a web admin panel and API capabilities to configure peering servers in order to terminate calls to non-local subscribers. Those peering servers may become *temporarily unavailable* due to overloading or networking issues. The Sipwise C5 will fail over to another peering server (matching the corresponding peering rules) after a timeout configured at system level (see the `sems.sbc.outbound_timeout` configuration parameter; 6 sec by default), if no provisional response (a response with a code in the range of 100 to 199) is received for the outbound INVITE request.

Even if this timer is set much lower, like 3 sec, the call setup time is increased significantly. This is even more true if multiple peering servers fail at the same time, which will sum up the individual timeouts, finally *causing call setup times reach the order of tens of seconds*.



To optimize the call setup time in such scenarios, a new feature is implemented to *continuously probe peering servers* via SIP messages, and mark them as unavailable on timeout or when receiving unexpected response codes. Appropriate SIP response codes from the peering servers will mark them as available again.

Peering servers *marked as unavailable* are then *skipped during call routing* in the peering selection process, which significantly shortens the call setup times if peering servers fail.

### 7.12.2 Configuration of Peer Probing

The system administrator has to configure the peer probing feature in 2 steps:

1. System-level configuration enables the peer probing feature in general on the Sipwise C5 and determines the operational parameters, such as timeouts, the SIP method used for probing requests, etc.
2. Peering server configuration will add / remove a peering server to the list of probed endpoints.

#### 7.12.2.1 System-level Configuration

The parameters of peer probing are found in the main system configuration file `/etc/ngcp-config/config.yml`. You can see the complete list of configuration parameters in Section [B.1.13](#) of the handbook, while the most significant ones are discussed here.

**Enabling peer probing** system-wide happens through the `kamailio.proxy.peer_probe.enable` parameter. If it is set to *yes* (which is the default value) then Sipwise C5 will consider probing of individual peering servers based on their settings.

**Timeout** of a single probing request can be defined through `kamailio.proxy.peer_probe.timeout` parameter. This is a value interpreted as seconds while Sipwise C5 will wait for a SIP response from the peering server. Default is 5 seconds.

The **probing interval** can be set through the `kamailio.proxy.peer_probe.interval` parameter. This is the time period in seconds that determines how often a probing request is sent to the peering servers. Default is 10 seconds.

The **SIP method** used for probing requests can be defined through `kamailio.proxy.peer_probe.method` parameter. Allowed values are: `OPTIONS` (default) and `INFO`.

---

#### Tip

The system administrator, in most of the cases, will not need to modify the default configuration values other than that of timeout and interval.

---

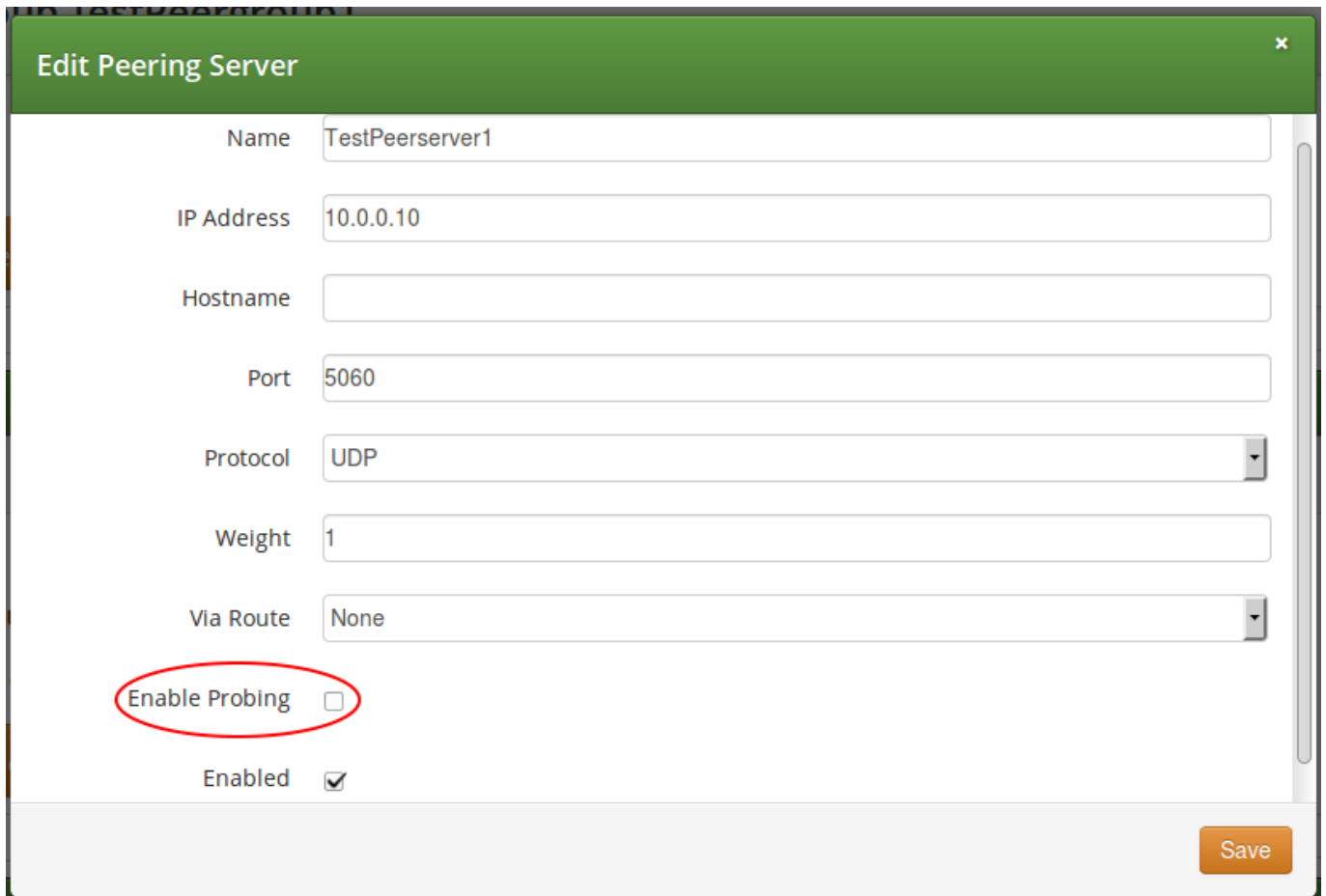
**If no available peering server is found, the call is rejected** with the response code and reason configured in `kamailio.proxy.early_rejects.peering_unavailable.announce_reason`. If a sound file is configured within the *system sound set* assigned to the calling party, an announcement is played as early media before the rejection.

#### 7.12.2.2 Individual Peering Server Configuration

When the peer probing feature is enabled on system-level, it is possible to add each individual peering server to the list of probed endpoints. You can change the probed status of a server in two ways:

**Enable probing of a peering server via the admin web interface**

1. Open the properties panel of a peering server: *Peerings* → *select a peering group* → *Details* → *select a peering server* → *Edit*
2. Tick the checkbox *Enable Probing*
3. Save changes



The screenshot shows a web form titled "Edit Peering Server". The form has a green header bar with the title and a close button. Below the header, there are several input fields and dropdown menus:

- Name: TestPeerserver1
- IP Address: 10.0.0.10
- Hostname: (empty)
- Port: 5060
- Protocol: UDP (dropdown)
- Weight: 1
- Via Route: None (dropdown)
- Enable Probing:  (circled in red)
- Enabled:

At the bottom right of the form, there is an orange "Save" button.

Figure 52: Enable Probing of Peering Server

**Enable probing of a peering server via the REST API**

- when you *create a new peering server* you will use an HTTP *POST* request and the target URL:  
[https://<IP\\_of\\_NGCP>:1443/api/peeringservers](https://<IP_of_NGCP>:1443/api/peeringservers)
- when you *update an existing peering server* you will use an HTTP *PUT* or *PATCH* request and the target URL:  
[https://<IP\\_of\\_NGCP>:1443/api/peeringservers/id](https://<IP_of_NGCP>:1443/api/peeringservers/id)

In all cases you have to set the *probe* property to `true` in order to enable probing, and to `false` in order to disable probing.

Default value is `false` and this property may be omitted in a create/update request, which ensures backward compatibility of the `/api/peeringservers` API resource.

### 7.12.3 Monitoring of Peer Probing

Peering server states, such as "reachable" / "unreachable", are continuously stored in a time-series database (InfluxDB type) by Sipwise C5 Proxy nodes. It is possible to **graphically represent the state of peering servers** on NGCP's admin web interface, just like other system variables (like CPU and memory usage, number of registered subscribers, etc.). However this is not available by default and must be configured by Sipwise.

### 7.12.4 Further Details for Advanced Users

---

#### Tip

This subchapter of the handbook is targeted on advanced system operators and Sipwise engineers and is not necessary to read in order to properly manage peer probing feature of NGCP.

---

#### 7.12.4.1 Behaviour of Kamailio Proxy Instances

Each *kamailio-proxy* instance on the proxy nodes performs the probing individually for performance reasons. Each proxy holds its result in its cache to avoid central storage and replication of the probing results.

Each peering server is cross-checked against the hash table filled during outbound probing requests and is skipped by call routing logic, if a match is found.

On start or restart of the *kamailio-proxy* instance, the probing will start after the first interval, and NOT immediately after start. In the first probing interval the proxy will always try to send call traffic to peering servers until the first probing round is finished, and will only then start to skip unavailable peering servers.

#### 7.12.4.2 Changes to Kamailio Proxy Configuration

A new configuration template: `/etc/ngcp/config/templates/etc/kamailio/proxy/probe.cfg.tt2` is introduced to handle outbound probing requests.

#### 7.12.4.3 Database Changes

A new DB column: `provisioning.voip_peer_hosts.probe` with type `TINYINT(1)` (boolean) is added to the DB schema.

A peer status change will populate the `kamailio.dispatcher` table, inserting the SIP URI in format `sip:$ip:$port;transport=...` in dispatcher group 100, which defines the probing group for peering servers.

Also the `kamailio.dispatcher.attrs` column is populated with a parameter `peerid=$id`. This ID is used during probing to load the peer preferences: `outbound_socket` and `lbrtp_set`, that are required to properly route the probing request.

## 7.13 Voicemail System

### 7.13.1 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*.

#### 7.13.1.1 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.

#### 7.13.1.2 External IVR access

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

You can globally configure a rewrite rule in `config.yml` using `asterisk.voicemail.normalize_match` and `asterisk.voicemail.normalize_replace` allowing you to customize the format a subscriber can enter, e.g. having `^0([1-9][0-9]+)$` as match part and `49$1` as replace part to accept German national format.

### 7.13.2 IVR Menu Structure

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

- 1 Read voicemail messages
  - 3 Advanced options
    - \* 3 To Hear messages Envelope
    - \* \* Return to the main menu
  - 4 Play previous message
  - 5 Repeat current message
  - 6 Play next message

- 7 Delete current message
- 9 Save message in a folder
  - \* 0 Save in new Messages
  - \* 1 Save in old Messages
  - \* 2 Save in Work Messages
  - \* 3 Save in Family Messages
  - \* 4 Save in Friends Messages
  - \* # Return to the main menu
- 2 Change folders
  - 0 Switch to new Messages
  - 1 Switch to old Messages
  - 2 Switch to Work Messages
  - 3 Switch to Family Messages
  - 4 Switch to Friends Messages
  - # Get Back
- 3 Advanced Options
  - \* To return to the main menu
- 0 Mailbox options
  - 1 Record your unavailable message
    - \* 1 accept it
    - \* 2 Listen to it
    - \* 3 Rerecord it
  - 2 Record your busy message
    - \* 1 accept it
    - \* 2 Listen to it
    - \* 3 Rerecord it
  - 3 Record your name
    - \* 1 accept it
    - \* 2 Listen to it
    - \* 3 Rerecord it
  - 4 Record your temporary greetings
    - \* 1 accept it / or re-record if one already exist
    - \* 2 Listen to it / or delete if one already exist
    - \* 3 Rerecord it

- 5 Change your password
- \* To return to the main menu
- \* Help
- # Exit

### 7.13.3 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.

#### 7.13.3.1 Unavailable Message

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

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

#### 7.13.3.2 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."

#### 7.13.3.3 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.

## 7.13.4 Folders

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

### 7.13.4.1 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.

## 7.13.5 Voicemail Languages Configuration

To add a new language or to change the pronunciation for an existing one, ensure that **mode=new** is defined in `/etc/ngcp-config/templates/etc/asterisk/say.conf.tt2`. Adjust the configuration in the same file using the manual in the beginning. Then, as usual, make the new configuration active.

## 7.13.6 Flowcharts with Voice Prompts

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

7.13.6.1 Listening to New Messages

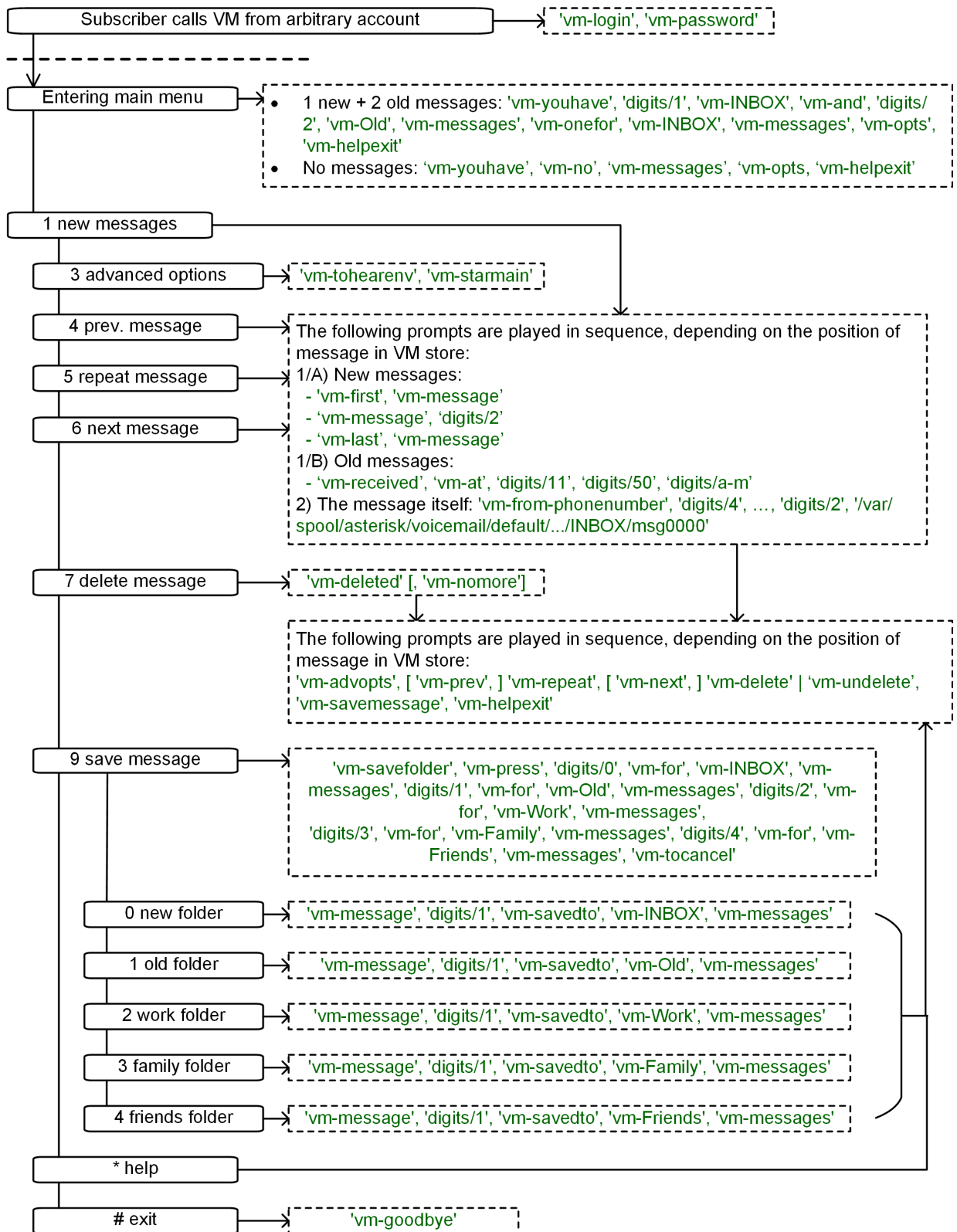


Figure 53: Flowchart of Listening to New Messages



7.13.6.2 Changing Voicemail Folders

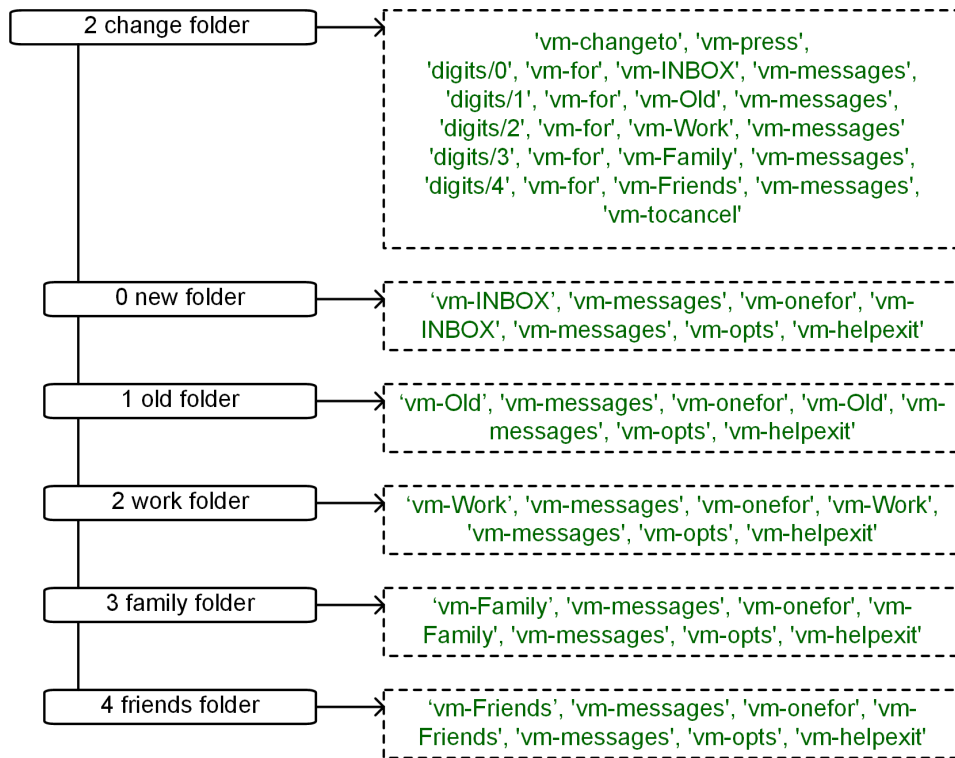


Figure 54: Flowchart of Changing Voicemail Folders

7.13.6.3 Mailbox Options

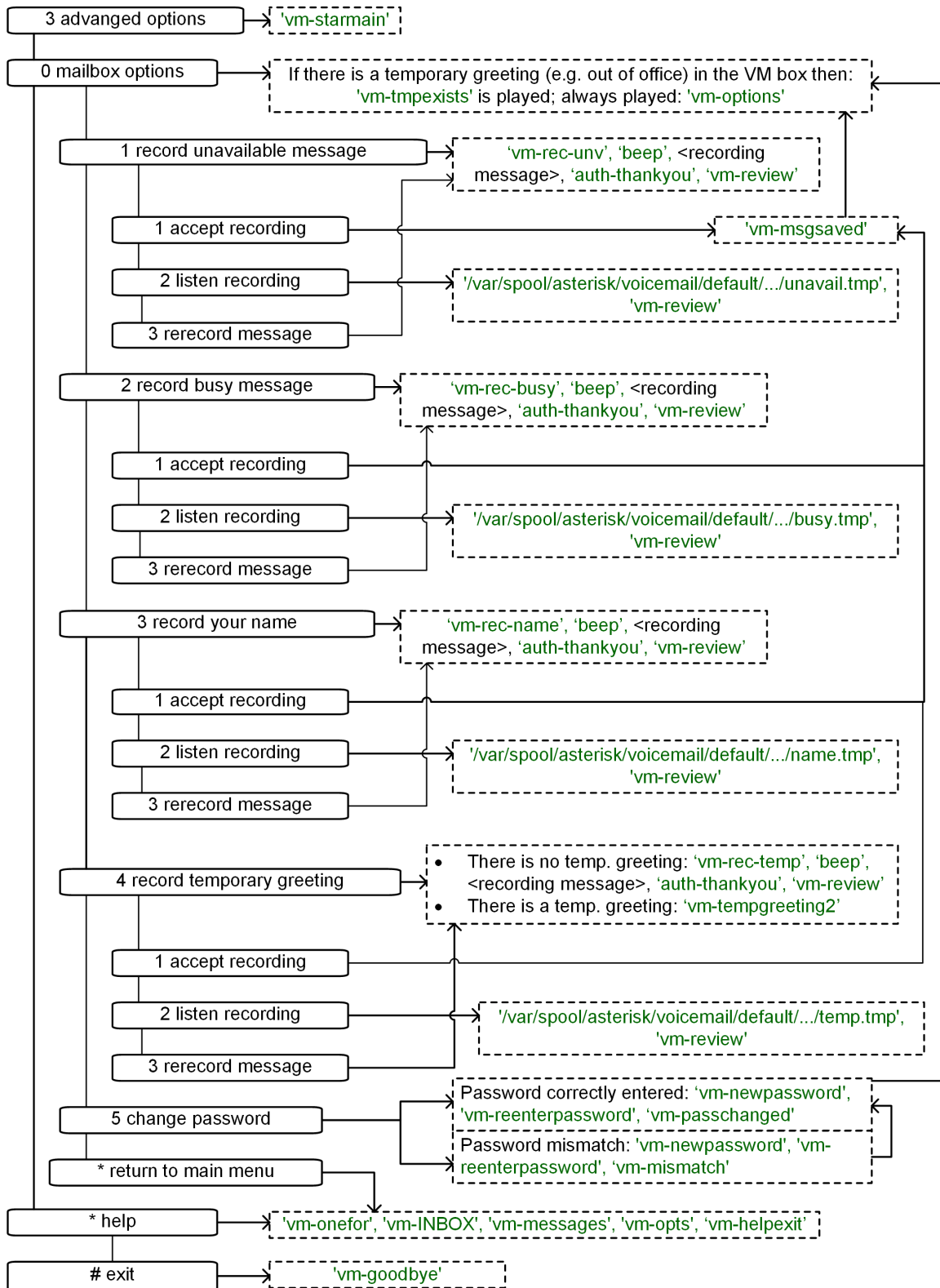


Figure 55: Flowchart of Changing Mailbox Options

7.13.6.4 Leaving a Message

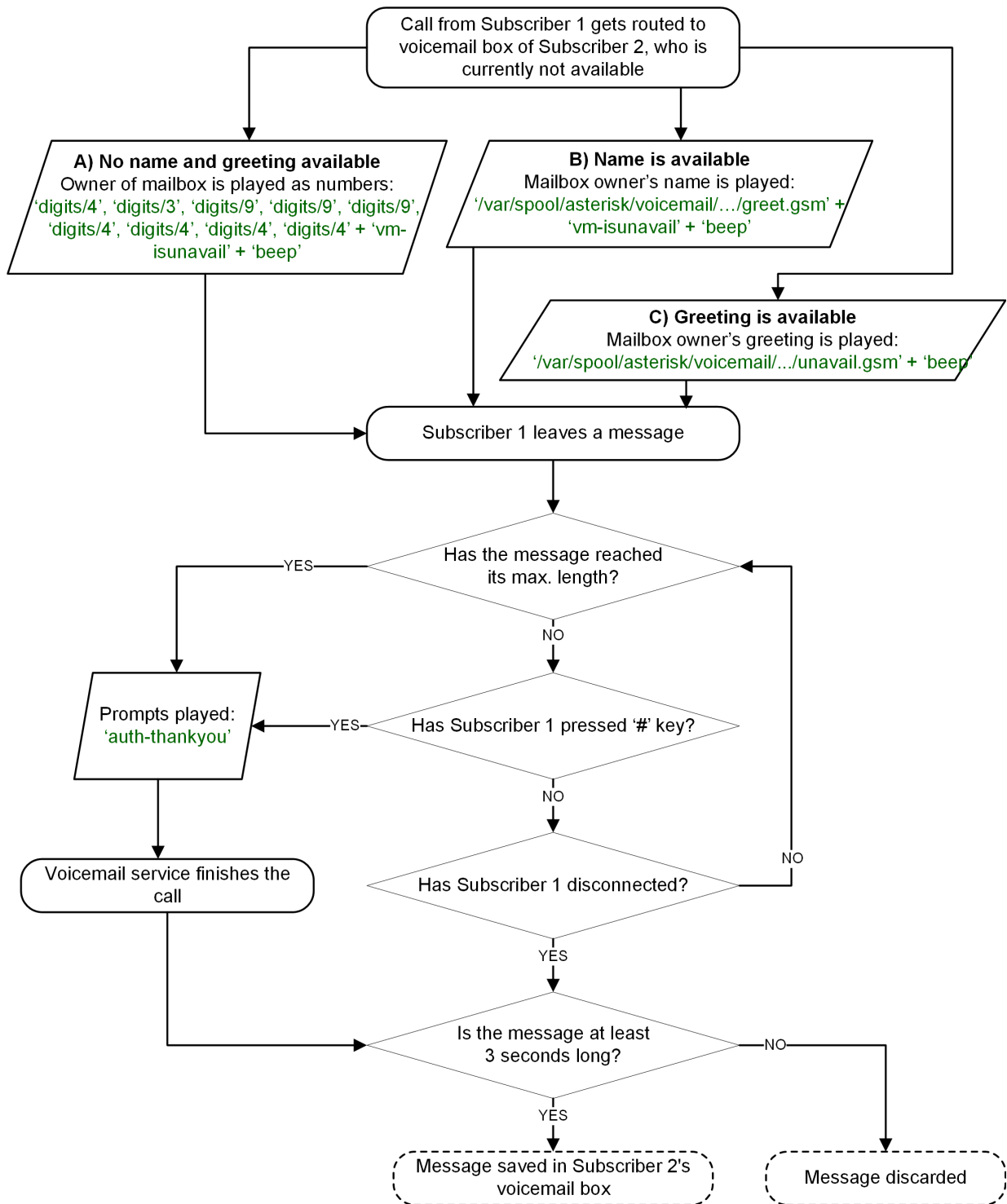
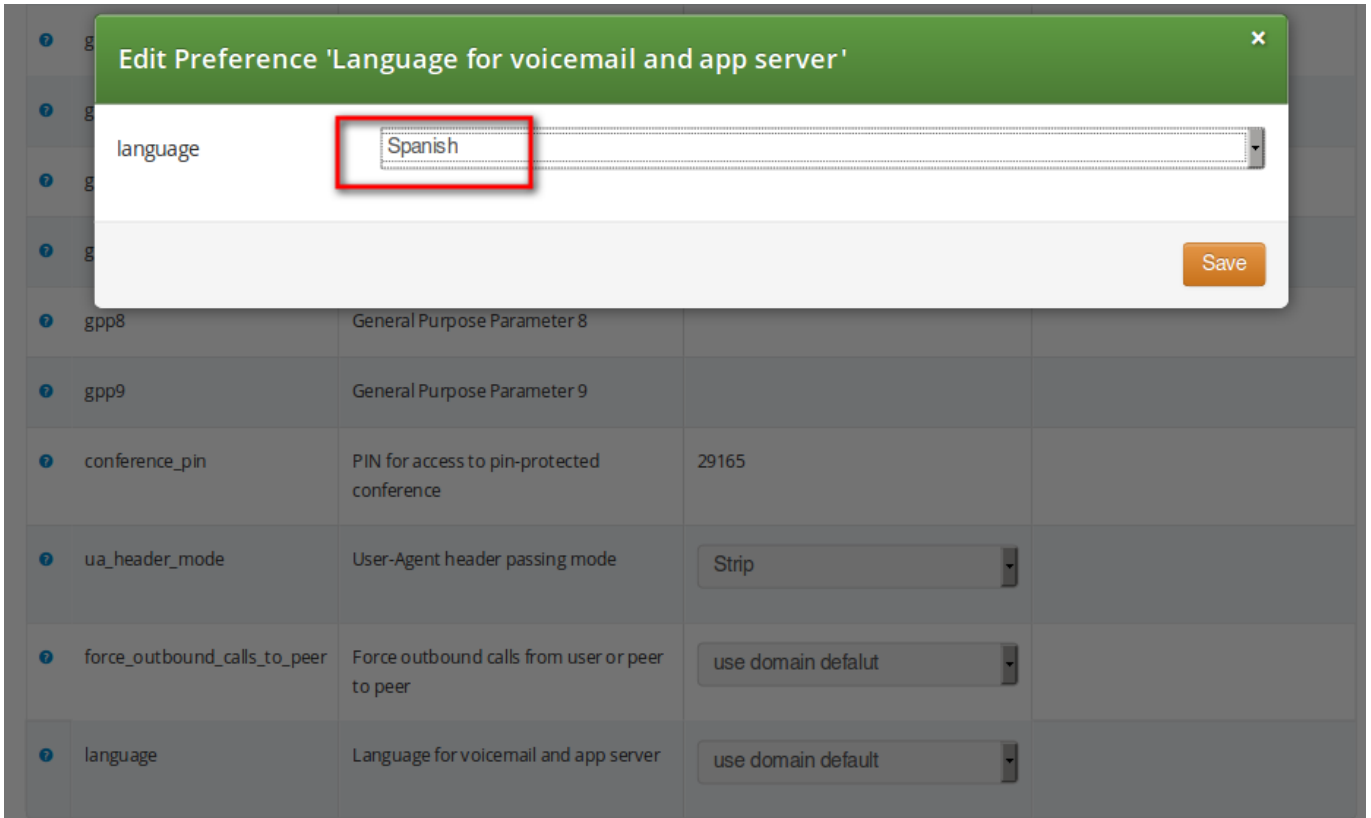


Figure 56: Flowchart of Leaving a Voice Message

## 7.14 Configuring Subscriber IVR Language

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



The Sipwise C5 provides the pre-installed prompts for the Voicemail in the English, Spanish, French and Italian languages and the pre-installed prompts for the Vertical Service Codes IVRs in English only.

The other IVRs such as the Conference system and the error announcements use the Sound Sets configured in Sipwise C5 Panel and uploaded by the administrator in his language of choice.

## 7.15 Sound Sets

The Sipwise C5 provides the administrator with ability to upload the voice prompts such as conference prompts or call error announcements on the *Sound Sets page*. There is a preference *sound\_set* in the *NAT and Media Flow Control* section on Domain and Subscriber levels to link subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one). Additionally Sound Sets can be configured on Peer level in order to play a dedicated early reject prompt when an incoming call doesn't match any local subscriber. Sound Sets can be defined in *Settings*→*Sound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button.

Logged in as administrator Language Logout

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

## Sound Sets

← Back ★ Create Sound Set

Sound set successfully created

Show 5 entries Search:

| # | Reseller | Customer | Name                | Description                       |   |
|---|----------|----------|---------------------|-----------------------------------|---|
| 1 | default  |          | Conference          |                                   | <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Files</a> |
| 2 | default  |          | Early media rejects | Failed call attempt announcements |   |

Showing 1 to 2 of 2 entries

**Note**

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

**7.15.1 Sound\_Set and Contract\_Sound\_Set Usage**

*Sound\_Set* and *Contract\_Sound\_Set* are used for different purposes:

- *Contract\_Sound\_Set* is a customer-specific sound set used for the Cloud PBX features. It can be assigned only to a PBX customer.
- *Sound\_Set* contains general prompts, e.g. for the Conference, Music on Hold, Early Rejects, etc.

The Music on Hold prompts can be uploaded to both *Contract\_Sound\_Set* and *Sound\_Set*. In this case, the one from the *Sound\_Set* will take precedence. Vice versa, digits prompts in *Contract\_Sound\_Set* will take precedence.

**7.15.2 Configuring Early Reject Sound Sets**

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

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

The call error announcements are played to the user in early media hence the name "Early Reject". If you don't provide the sound files for any handles they will not be used and Sipwise C5 will fallback to sending the error response code back to the user.

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

Table 5: Early Reject Announcements

| Handle                                  | Description   | Message played                   |
|---|---|----------------------------------|
| <code>announce_before_call_setup</code> | This is an announcement that the calling party hears before the call is being actually sent to the destination.<br>The feature can be activated with Applications / <code>play_announce_before_call_setup</code> domain or subscriber preference. Loopplay doesn't have effect on this element. | N/A (custom message, no default) |
| <code>announce_before_cf</code>         | This is an announcement that the calling party hears before the call is being forwarded (Unconditional and Not Available cases) to the destination. The feature can be activated with Applications / <code>play_announce_before_cf</code> domain or subscriber preference.                      | N/A (custom message, no default) |

Table 5: (continued)

| Handle                    | Description   | Message played   |
|---------------------------|---|--|
| announce_before_recording | This is an announcement that the calling party hears before the call is actually sent to the destination and before the recording starts. The feature can be activated with Applications / play_announce_before_recording domain or subscriber preference. NAT and Media Flow Control / record_call preference has to be activated as well. Loopplay doesn't have effect on this element. | N/A (custom message, no default)   |
| block_in                  | This is what the calling party hears when a call is made from a number that is blocked by the incoming block list ( <i>adm_block_in_list</i> , <i>block_in_list</i> customer/subscriber preferences)  | Your call is blocked by the number you are trying to reach.                            |
| block_out                 | This is what the calling party hears when a call is made to a number that is blocked by the outgoing block list ( <i>adm_block_out_list</i> , <i>block_out_list</i> customer/subscriber preferences)  | Your call to the number you are trying to reach is blocked.                            |
| block_ncos                | This is what the calling party hears when a call is made to a number that is blocked by the NCOS level assigned to the subscriber or domain (the NCOS level chosen in <i>ncos</i> and <i>adm_ncos</i> preferences). <i>PLEASE NOTE</i> : It is not possible to configure the status code and text.  | Your call to the number you are trying to reach is not permitted.                      |
| block_override_pin_wrong  | Announcement played to calling party if it used wrong PIN code to override the outgoing user block list or the NCOS level for this call (the PIN set by <i>block_out_override_pin</i> and <i>adm_block_out_override_pin</i> preferences)  | The PIN code you have entered is not correct.  |
| callee_busy               | Announcement played on incoming call to the subscriber which is currently busy (486 response from the UAS)  | The number you are trying to reach is currently busy. Please try again later.          |
| callee_offline            | Announcement played on incoming call to the subscriber which is currently not registered  | The number you are trying to reach is currently not available. Please try again later. |

Table 5: (continued)

| Handle                                 | Description   | Message played   |
|--|---|--|
| <code>callee_tmp_unavailable</code>    | Announcement played on incoming call to the subscriber which is currently unavailable (408, other 4xx or no response code or 30x with malformed contact)  | The number you are trying to reach is currently not available. Please try again later.     |
| <code>callee_unknown</code>            | Announcement that is played on call to unknown or invalid number (not associated with any of our subscribers/hunt groups)   | The number you are trying to reach is not in use.  |
| <code>cf_loop</code>                   | Announcement played when the called subscriber has the call forwarding configured to itself   | The number you are trying to reach is forwarded to an invalid destination.                 |
| <code>emergency_geo_unavailable</code> | Announcement played when emergency destination is dialed but the destination is not provisioned for the location of the user.<br><i>PLEASE NOTE:</i> The configuration entry for this case in <code>/etc/ngcp-config/config.yml</code> file is <code>emergency_invalid</code> . | The emergency number you have dialed is not available in your region.                      |
| <code>emergency_unsupported</code>     | Announcement played when emergency destination is dialed but the emergency calls are administratively prohibited for this user or domain ( <i>reject_emergency</i> preference is enabled)   | You are not allowed to place emergency calls from this line. Please use a different phone. |
| <code>error_please_try_later</code>    | Announcement played when the call is handled by 3rd party call control (PCC) and there was an error during call processing.<br><i>PLEASE NOTE:</i> This announcement may be configured in the sound set in <code>voucher_recharge</code> section.                               | An error has occurred. Please try again later.   |
| <code>invalid_speeddial</code>         | This is what the calling party hears when it calls an empty speed-dial slot   | The speed dial slot you are trying to use is not available.                                |
| <code>locked_in</code>                 | Announcement played on incoming call to a subscriber that is locked for incoming calls  | The number you are trying to reach is currently not permitted to receive calls.            |
| <code>locked_out</code>                | Announcement played on outgoing call to subscriber that is locked for outgoing calls  | You are currently not allowed to place outbound calls.                                     |



Table 5: (continued)

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

Table 5: (continued)

| Handle               | Description   | Message played  |
|----------------------|---|---|
| voicebox_unavailable | <i>PLEASE NOTE:</i> This announcement is already obsolete, as of Sipwise C5 version mr5.3 | The voicemail of the number you are trying to reach is currently not available. Please try again later. |

There are some early reject scenarios when either **no voice announcement is played, or a fixed announcement is played**. In either case a SIP error status message is sent from Sipwise C5 to the calling party. It is possible to configure the exact status code and text for such cases in the `/etc/ngcp-config/config.yml` file, in `kamailio.proxy.early_rejects` section. The below table gives an overview of those early reject cases.

Table 6: Additional Early Reject Reason Codes

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

### 7.15.3 Play an announcement on behalf of callee server failure in case of outbound calls

The Sipwise C5 makes it possible to play an announcement on behalf of callee server failure in case of outbound calls. The features can be activated on Subscribers and on Peers. For example: if subscriber A calls subscriber B and B refuses the call with code 404 without providing any announcement, Sipwise C5 can be configured to play a customized announcement to A on behalf of B.

To activate this feature, first create a system *Sound Set*, or use an already existing one, and then assign it to the callee subscriber. Upload in the *Sound Set* one or more announcements. Once the *Sound Set* is configured, the subscriber's preference *announce\_error\_codes\_enable* must be enabled under *Subscriber* → *Preferences* → *NAT and Media Flow Control* menu. Last step is to list in the subscriber's preference *announce\_error\_codes\_list* the announcements that will be played to the caller in case a particular error code is returned back from the callee. Each entry of the list has to be a string composed in the following way: `<error_code>;<announcement_name>`, where *error\_code* is the SIP return code and *announcement\_name* is name of the announcement taken from the *sound\_set* list. Returning to the example above, to play *callee\_unknown* message in case of 404 returned from the callee, the entry `404;callee_unknown` has to be added in *announce\_error\_codes\_list* preference.

The same feature is available for peer as well.



#### Important

In case *announce\_error\_codes\_enable* is enabled, it is important that the remote endpoint doesn't play any announcement for error codes listed in *announce\_error\_codes\_list* otherwise the final result will be to have two announcements: one generated by the remote endpoint and one generated by Sipwise C5.

---

## 7.16 Conference System

The Sipwise C5 provides the simple pin-protected conferencing service built using the SEMS DSM scripting language. Hence it is open for all kinds of modifications and extensions.

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

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

### 7.16.1 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*).

Destination  Voicemail  Conference  URI/Number

URI/Number

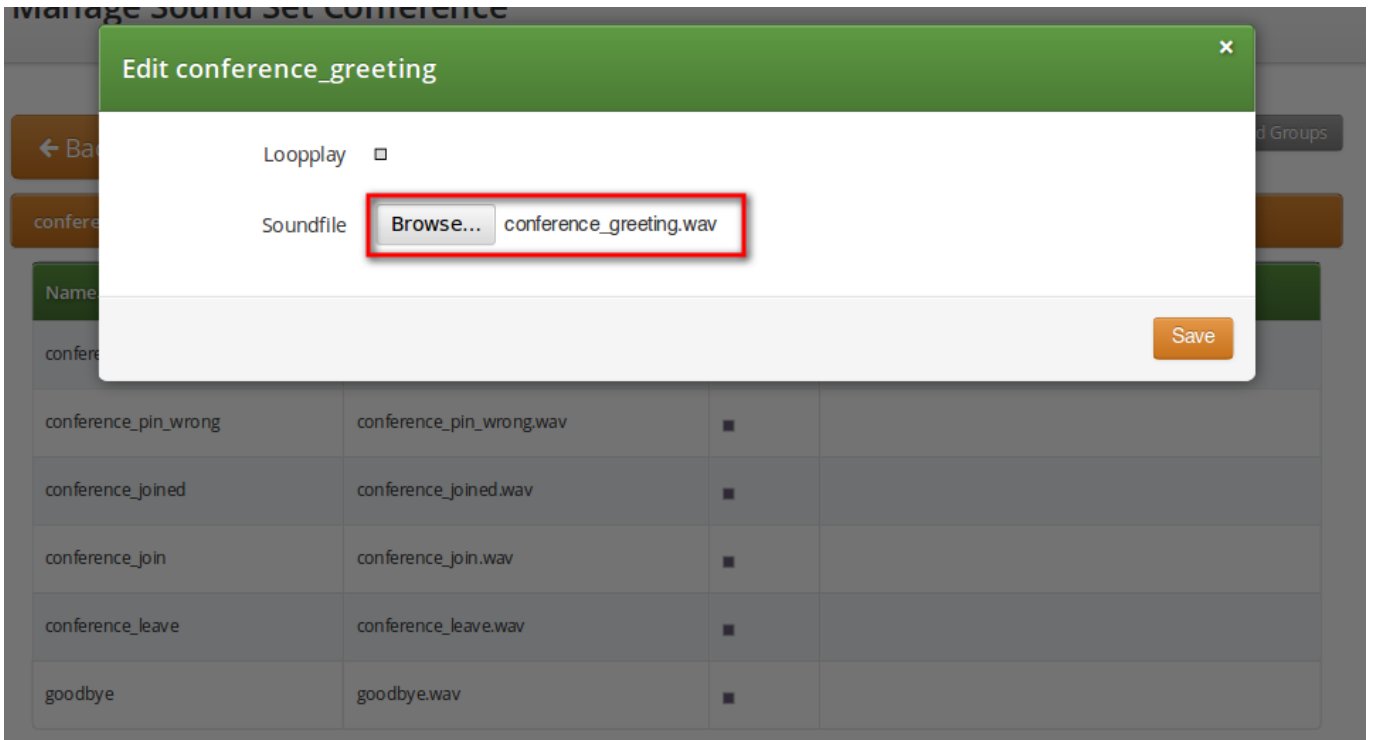
for (seconds)

Advanced View Save

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

### 7.16.2 Configuring Conference Sound Sets

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



Upload the following files:

Table 7: 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).

### 7.16.3 Joining the Conference

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

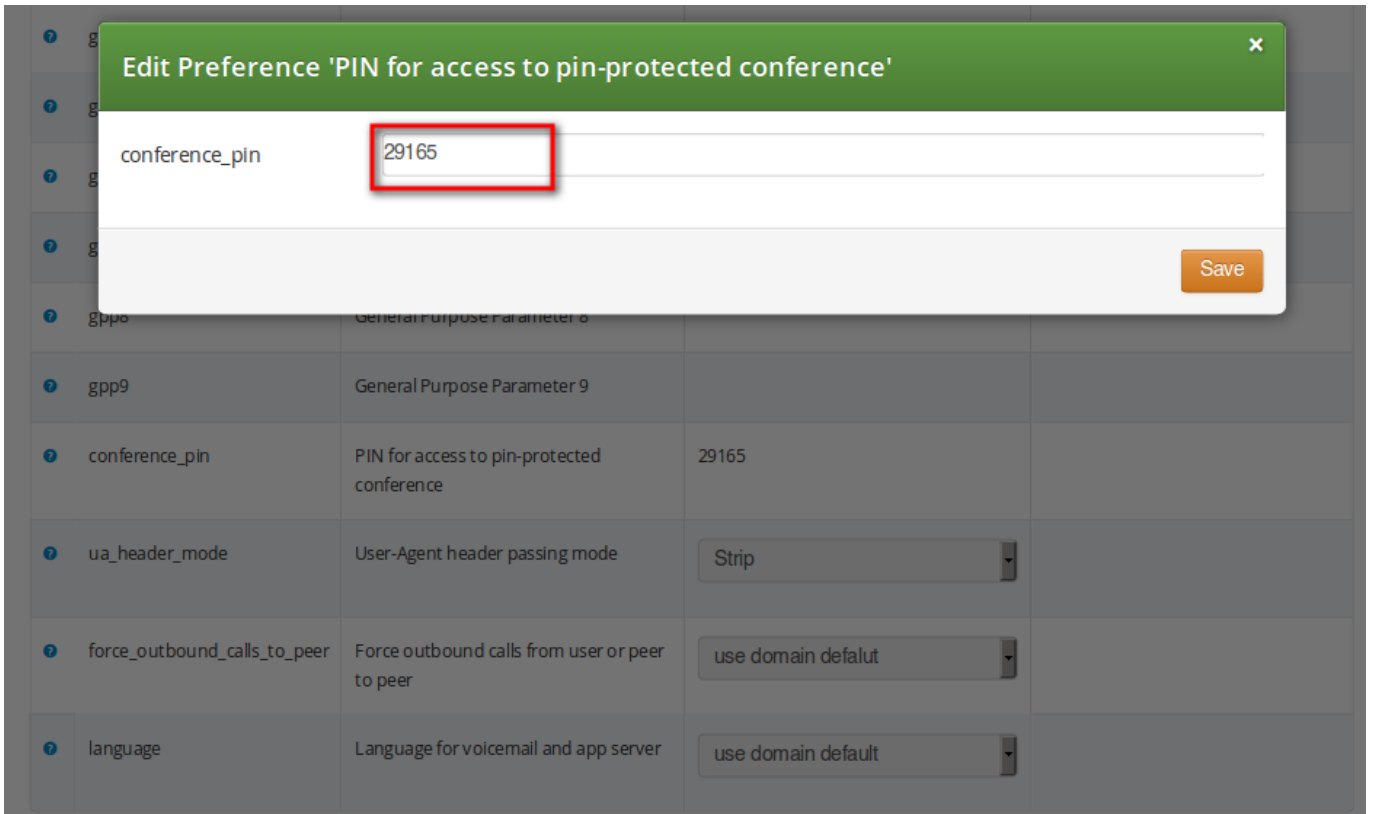


Figure 57: Setting Conference PIN

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

### 7.16.4 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.

7.16.4.1 Conference Flowchart with PIN Validation

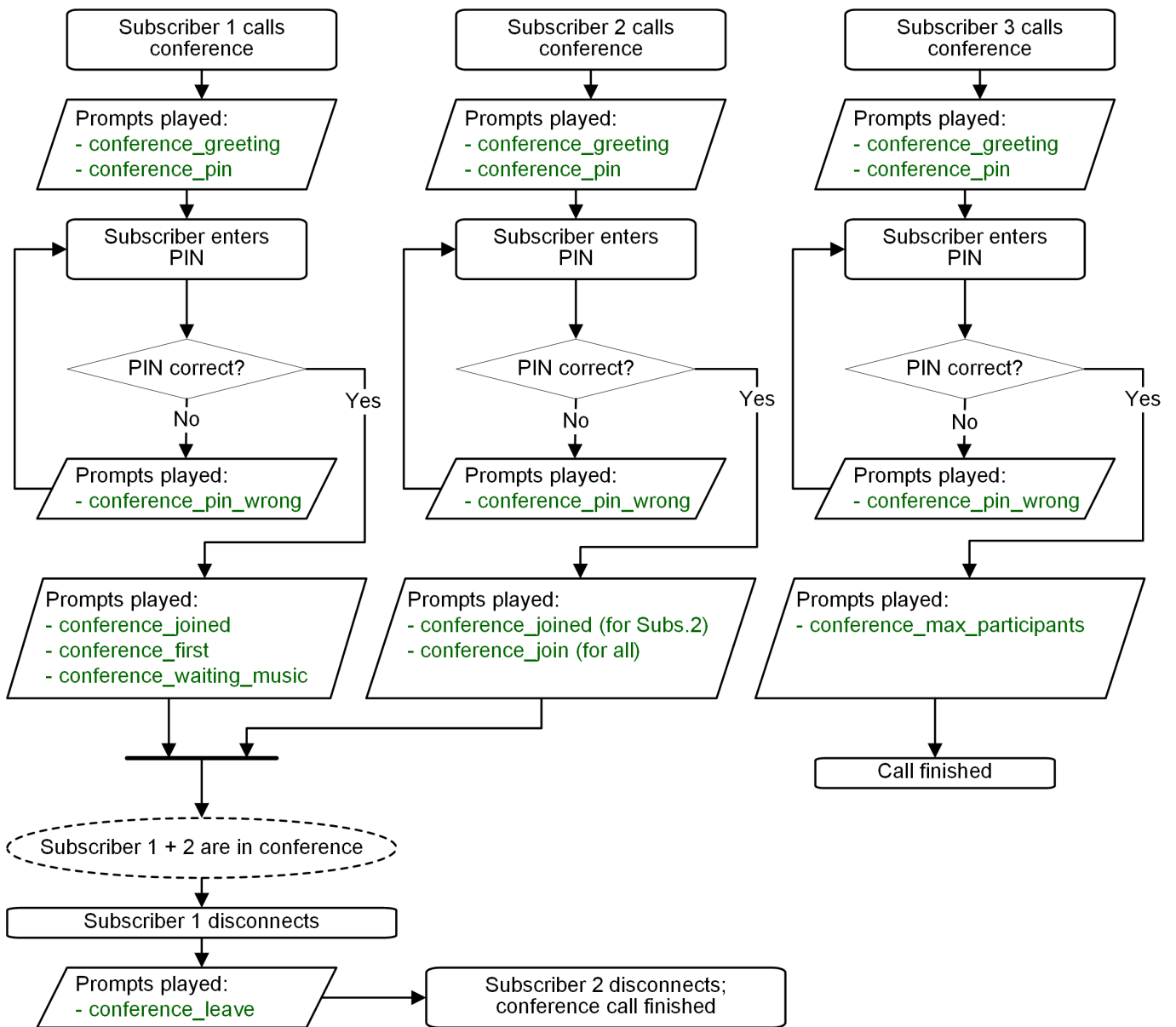


Figure 58: Flowchart of Conference with PIN Validation

**7.16.4.2 Conference Flowchart without PIN**

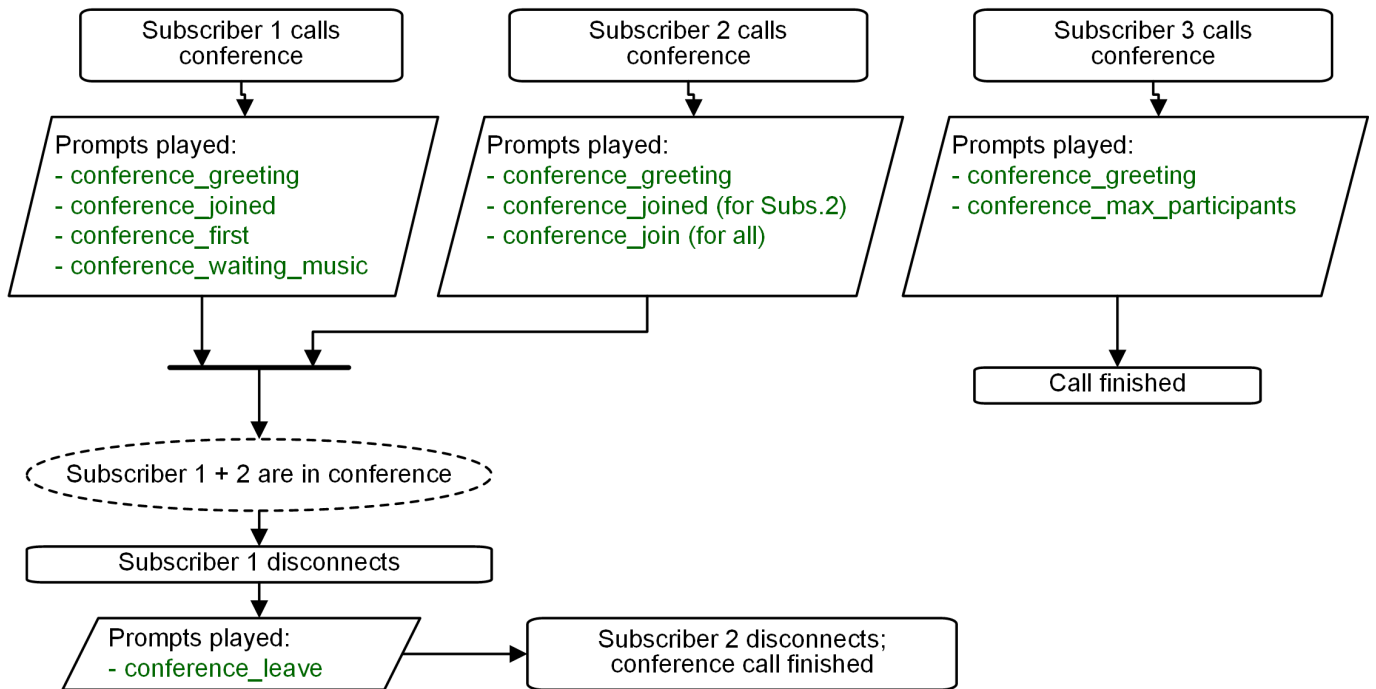


Figure 59: Flowchart of Conference without PIN

**7.17 Malicious Call Identification (MCID)**

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

**7.17.1 Setup**

To enable the feature first edit `config.yml` and enable there `apps: malicious_call: yes` and `kamailio: store_recent_calls: yes`. The latter option enables kamailio to store recent calls per subscriber UUID in the redis DB (the amount of stored recent calls will not exceed the amount of provisioned subscribers).

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

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

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

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



### 7.17.2 Usage

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

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

## 7.18 Subscriber Profiles

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

### 7.18.1 Subscriber Profile Sets

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

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

The screenshot displays the Sipwise NGCP Dashboard interface. At the top, it shows the user is logged in as an administrator, with options for language and logout. The dashboard title is 'sip:wise NGCP Dashboard'. A modal window titled 'Create Subscriber Profile Sets' is open, showing a list of resellers and form fields for creating a new profile set.

Reseller Search:

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

Showing 1 to 2 of 2 entries

Name:

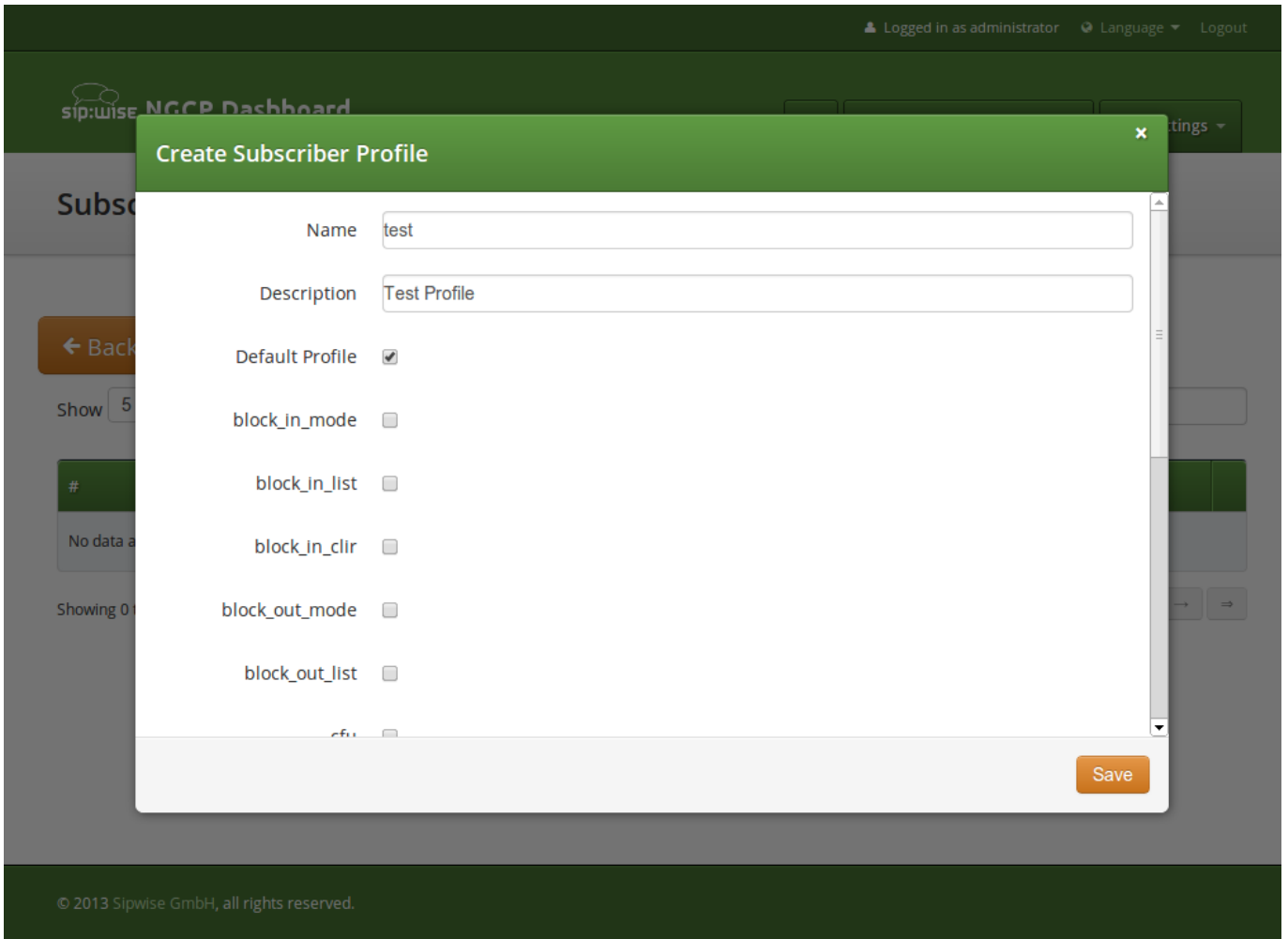
Description:

© 2013 Sipwise GmbH, all rights reserved.

You need to provide a reseller, name and description.

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

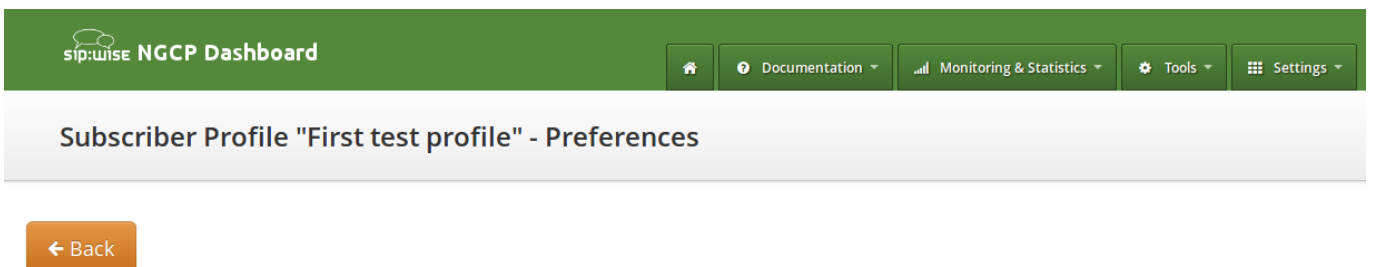
Profiles within a Profile Set can be created by clicking the *Create Subscriber Profile* button.



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

**Note**

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



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

← Back Expand Groups

Call Blockings

|   | Attribute | Name       | Value                        |
|---|-----------|------------|------------------------------|
| + | ncos      | NCOS Level | Test NCOS for blocking outca |

## 7.19 SIP Loop Detection

In order to detect a SIP loop ( incoming call as a response for a call request ) Sipwise C5 checks the combination of *SIP-URI*, *To* and *From* headers.

This check can be enabled in config.yml by setting `kamailio.proxy.loop_detection.enable: yes`. The system tolerates `kamailio.proxy.loop_detection.expire` seconds. Higher occurrence of loops will be reported with a SIP 482 "Loop Detected" error message

## 7.20 Invoices and Invoice Templates

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

---

### Note

The Sipwise C5 generates invoices in pdf format.

---

### 7.20.1 Invoices Management

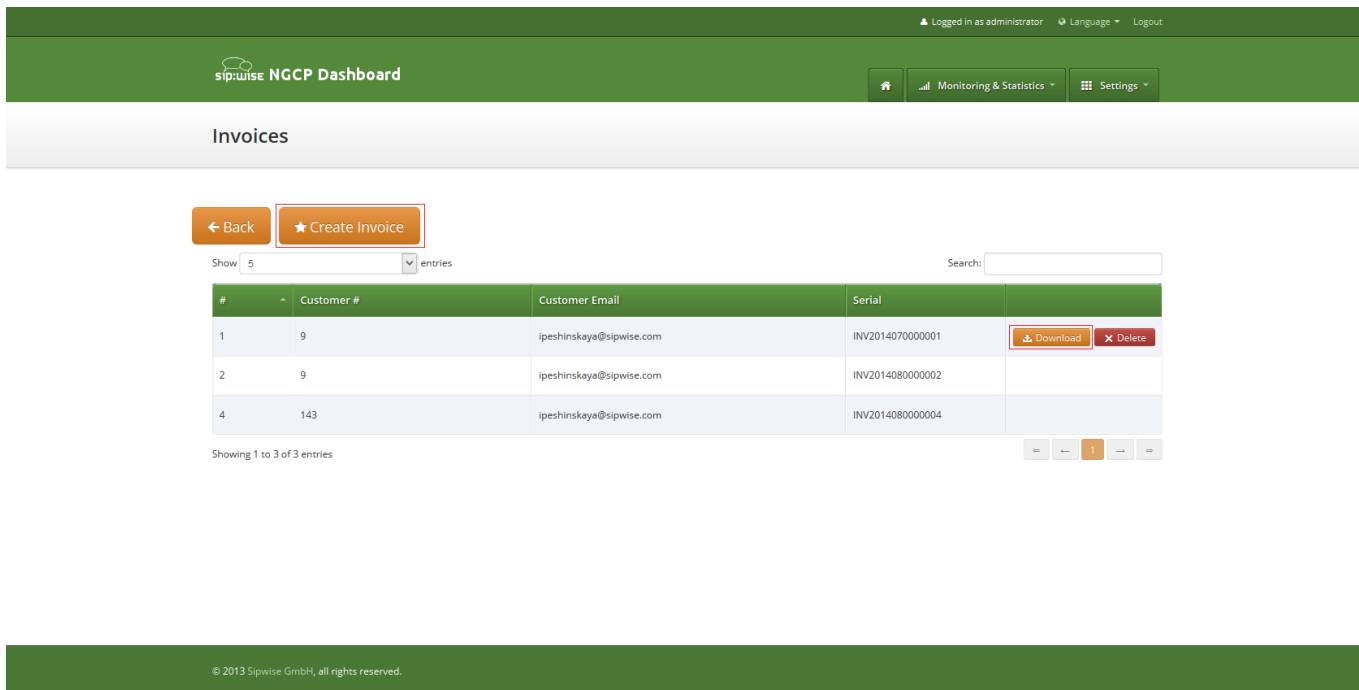
Invoices can be requested for generation, searched, downloaded and deleted on the administrative web interface. Navigate to *Settings* → *Invoices* menu and you get a list of all invoices currently stored in the database.

---

### Tip

The system operator or a third party application can also generate, list, retrieve and delete invoices via the REST API. Please read further details [here](#).

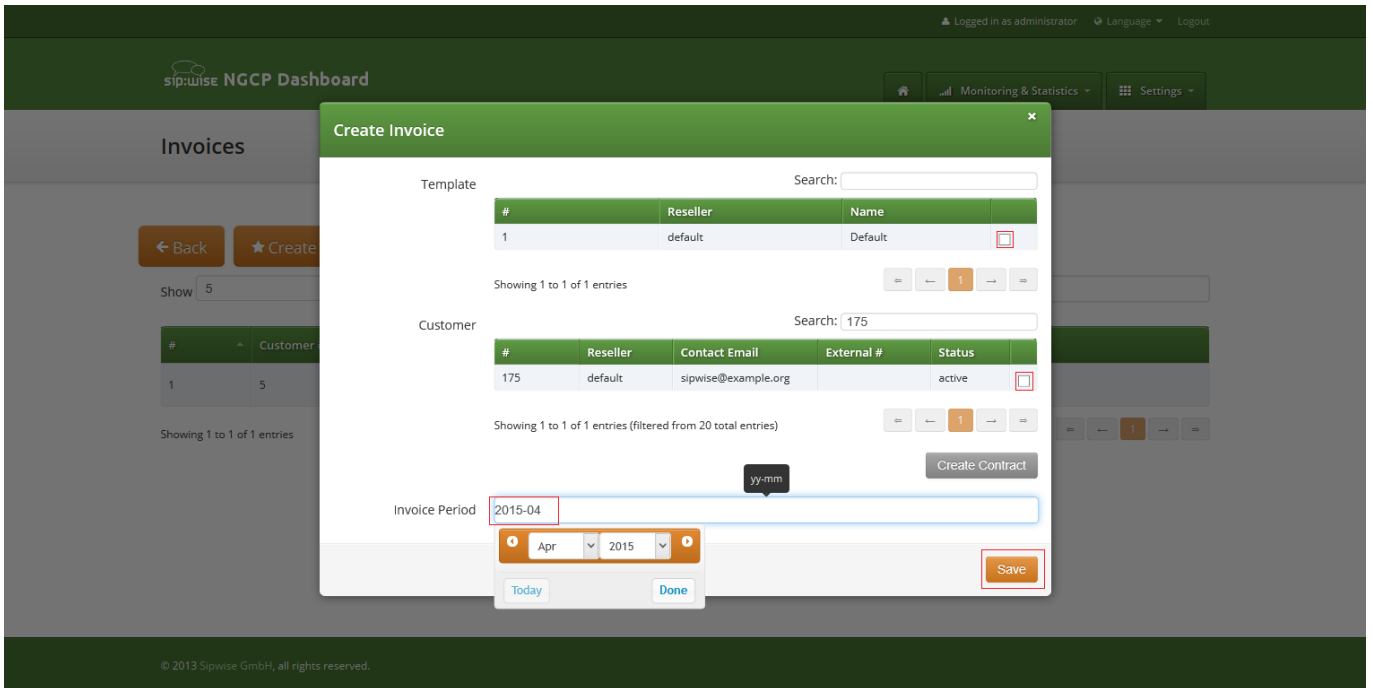
---



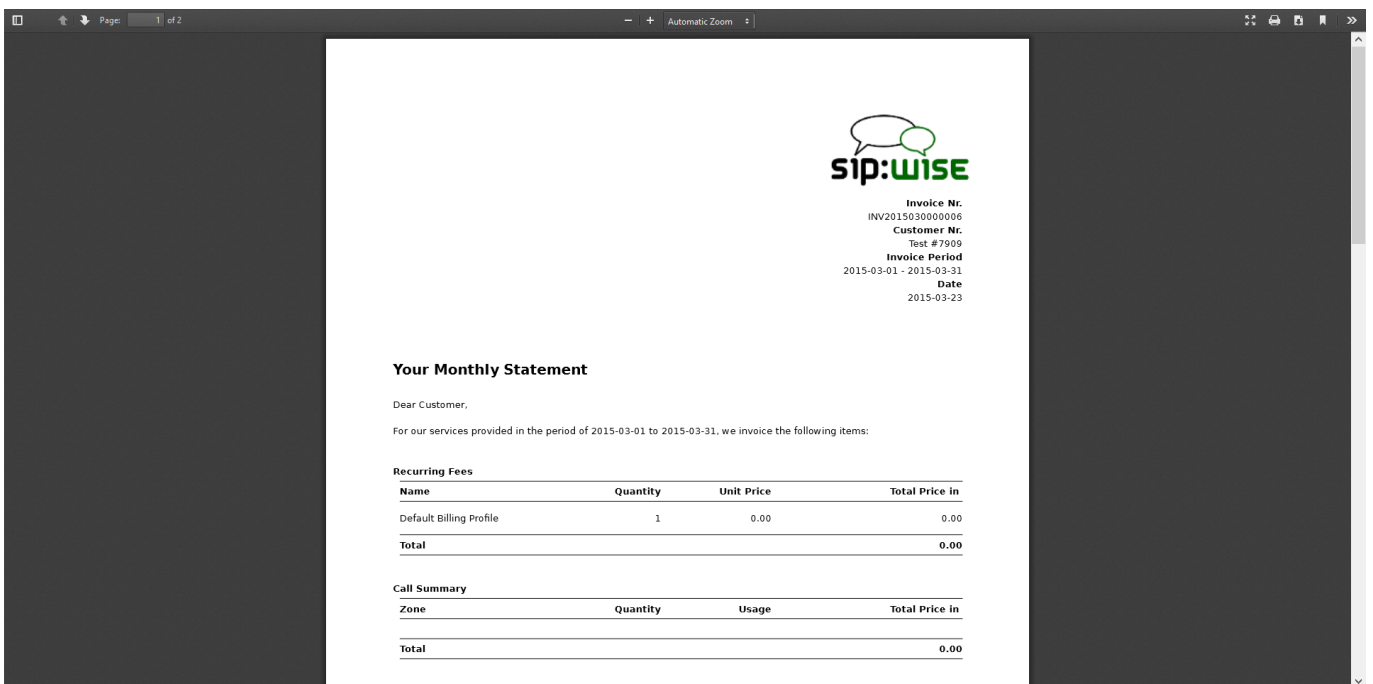
To request invoice generation for the particular customer and period press "Create invoice" button. On the invoice creation form following parameters are available for selection:

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

All form fields are mandatory.



Generated invoice can be downloaded as pdf file.



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

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

### 7.20.2 Invoice Management via REST API

Besides managing invoices on the admin web interface of NGCP, the system administrator (or a third party system) has the opportunity to request generation and retrieval of invoices via the *REST API*.

The subsequent sections describe the available operations for invoice management with API requests in details. All operations work on the *Invoices* resource and use the `/api/invoices` base path. The authentication method is username/password in the examples given below, however it is recommended to use a TLS client certificate for authentication on the REST API.

---

**Note**

The full API documentation is always available at the location: `https://<IP_of_NGCP_web_panel>:1443/api`

---

### 7.20.2.1 Generate a New Invoice

The **prerequisite** for generating a new invoice is that the customer has to have **an invoice template** assigned to him.

The following example shows a CURL command that will request generation of an invoice:

- for customer with ID "79"
- for the time period of November 2017
- based on the invoice template with ID "1"

```
curl -i -X POST -H 'Connection: close' -H 'Content-Type: application/json' \
--user adminuser:adminpwd -k 'https://127.0.0.1:1443/api/invoices/' \
--data-binary '{ "customer_id" : "79", "template_id" : "1", \
"period_start": "2017-11-01 00:00:00", "period_end": "2017-11-30 23:59:59" }'
```

Please note that in this operation we used the `/api/invoices` path (the *invoices* collection) and a *POST* request on it to create a new invoice item.

In case of a **successful operation**, Sipwise C5 will reply with **201 Created** HTTP status and send the ID of the invoice in *Location* header. In our example the new invoice item may be directly referred as `/api/invoices/3` (ID = 3).

```
HTTP/1.1 201 Created
Server: nginx
Date: Tue, 14 Nov 2017 13:38:40 GMT
Content-Length: 0
Connection: close
Location: /api/invoices/3
Set-Cookie: ngcp_panel_session=d5e4a8dd003fd7cac646653a6b5aefa703cf3e66; path=/; expires= ↵
Tue, 14-Nov-2017 14:38:38 GMT; HttpOnly
X-Catalyst: 5.90114
Strict-Transport-Security: max-age=15768000
```

In case of a **failed operation**, e.g. when we request an invoicing period that is invalid for the customer, Sipwise C5 will reply with **422 Unprocessable Entity** or **500 Internal Server Error** HTTP status.

### 7.20.2.2 Download Invoice Data

You can download properties / data of a specific invoice by selecting the item by its ID, using an HTTP *GET* request.

```
curl -i -X GET -H 'Connection: close' --user adminuser:adminpwd -k \  
'https://127.0.0.1:1443/api/invoices/3'
```

The above request will return a JSON data structure containing invoice properties:

```
HTTP/1.1 200 OK  
Server: nginx  
Date: Wed, 15 Nov 2017 12:13:04 GMT  
Content-Type: application/hal+json; profile="http://purl.org/sipwise/ngcp-api/"; charset= ↵  
utf-8  
Content-Length: 759  
Connection: close  
Link: </api/invoices/>; rel=collection  
Link: <http://purl.org/sipwise/ngcp-api/>; rel=profile  
Link: </api/invoices/3>; rel="item self"  
Link: </api/invoices/3>; rel="item http://purl.org/sipwise/ngcp-api/#rel-invoices"  
Link: </api/customers/79>; rel="item http://purl.org/sipwise/ngcp-api/#rel-customers"  
Set-Cookie: ngcp_panel_session=219fecbbe4fa936defdlee511c84efe7b5a6d6a; path=/; expires= ↵  
Wed, 15-Nov-2017 13:13:03 GMT; HttpOnly  
Strict-Transport-Security: max-age=15768000  
  
{  
  "_links" : {  
    "collection" : {  
      "href" : "/api/invoices/"  
    },  
    "curies" : {  
      "href" : "http://purl.org/sipwise/ngcp-api/#rel-{rel}",  
      "name" : "ngcp",  
      "templated" : true  
    },  
    "ngcp:customers" : {  
      "href" : "/api/customers/79"  
    },  
    "ngcp:invoices" : {  
      "href" : "/api/invoices/3"  
    },  
    "profile" : {  
      "href" : "http://purl.org/sipwise/ngcp-api/"  
    },  
    "self" : {
```



```

    "href" : "/api/invoices/3"
  }
},
"amount_net" : 0,
"amount_total" : 0,
"amount_vat" : 0,
"id" : 3,
"period_end" : "2017-11-30T23:59:59+00:00",
"period_start" : "2017-11-01T00:00:00+00:00",
"sent_date" : null,
"serial" : "INV2017110000003"
}

```

It is also possible to query the complete *invoices* collection and use a filter (e.g. invoicing period, customer ID, etc.) to get the desired invoice item. In the example below we request all available invoices that belong to the customer with ID "79".

```

curl -i -X GET -H 'Connection: close' --user adminuser:adminpwd -k \
'https://127.0.0.1:1443/api/invoices/?customer_id=79'

```

The returned dataset is now slightly different because it is represented as an array of items, although in our example the array consist of only 1 item:

```

{
  "_embedded" : {
    "ngcp:invoices" : [
      {
        "_links" : {
          "collection" : {
            "href" : "/api/invoices/"
          },
          "curies" : {
            "href" : "http://purl.org/sipwise/ngcp-api/#rel-{rel}",
            "name" : "ngcp",
            "templated" : true
          },
          "ngcp:customers" : {
            "href" : "/api/customers/79"
          },
          "ngcp:invoices" : {
            "href" : "/api/invoices/3"
          },
          "profile" : {
            "href" : "http://purl.org/sipwise/ngcp-api/"
          },
          "self" : {

```

```

        "href" : "/api/invoices/3"
      }
    },
    "amount_net" : 0,
    "amount_total" : 0,
    "amount_vat" : 0,
    "id" : 3,
    "period_end" : "2017-11-30T23:59:59+00:00",
    "period_start" : "2017-11-01T00:00:00+00:00",
    "sent_date" : null,
    "serial" : "INV2017110000003"
  }
]
},
"_links" : {
  "curies" : {
    "href" : "http://purl.org/sipwise/ngcp-api/#rel-{rel}",
    "name" : "ngcp",
    "templated" : true
  },
  "ngcp:invoices" : {
    "href" : "/api/invoices/3"
  },
  "profile" : {
    "href" : "http://purl.org/sipwise/ngcp-api/"
  },
  "self" : {
    "href" : "/api/invoices/?page=1&rows=10"
  }
},
"total_count" : 1
}

```

### 7.20.2.3 Download Invoice as PDF File

You can download a specific invoice as a PDF file in the following way:

- selecting the item by its ID (as in our example, but you can also use a filter and query the complete *invoices* collection)
- using an HTTP *GET* request
- adding "**Accept: application/pdf**" header to the request

```

curl -X GET -H 'Connection: close' -H 'Accept: application/pdf' \
  --user adminuser:adminpwd -k 'https://127.0.0.1:1443/api/invoices/3' > result.pdf

```

**Please note** that in the example above we **do not add the "-i" option** that would also include the headers of the HTTP response in the output file. The output of the CURL command, i.e. the PDF file, is saved as "result.pdf" locally.

#### 7.20.2.4 Delete an Invoice

In order to delete an invoice item you have to send a *DELETE* request on the specific item:

```
curl -i -X DELETE -H 'Connection: close' --user adminuser:adminpwd -k \  
'https://127.0.0.1:1443/api/invoices/3'
```

In case of successful deletion Sipwise C5 should send HTTP status 204 No Content as a response:

```
HTTP/1.1 204 No Content  
Server: nginx  
Date: Wed, 15 Nov 2017 13:42:42 GMT  
Connection: close  
Set-Cookie: ngcp_panel_session=10b66a6baf25a09739c2bb2377c70eccee78387; path=/; expires= ↵  
Wed, 15-Nov-2017 14:42:42 GMT; HttpOnly  
X-Catalyst: 5.90114  
Strict-Transport-Security: max-age=15768000
```

#### 7.20.3 Invoice Templates

Invoice template defines structure and look of the generated invoices. The Sipwise C5 makes it possible to create some invoice templates. Multiple invoice templates can be used to send invoices to the different customers using different languages.



##### Important

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

---

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

##### 7.20.3.1 Invoice Templates Management

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

Logged in as administrator | Language | Logout

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

## Invoice Templates

← Back | ★ Create Invoice Template

Search:

| # | Reseller | Name    | Type |   |
|---|----------|---------|------|---|
| 1 | default  | Default | svg  | <a href="#">Edit Meta</a> <a href="#">Edit Content</a> <a href="#">Delete</a> |

Showing 1 to 1 of 1 entries

© 2013 Sipwise GmbH, all rights reserved.

Invoice template creation is separated on two steps:

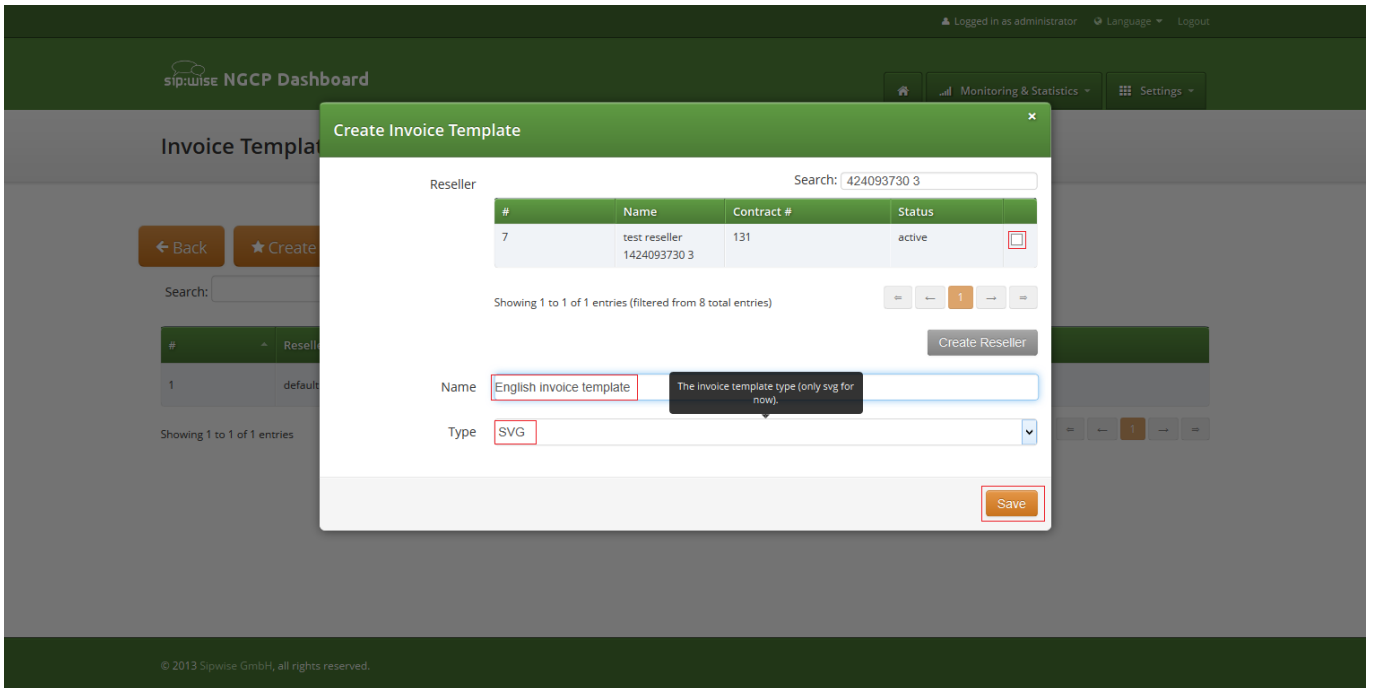
- Register new invoice template meta information.
- Edit content (template itself) of the invoice template.

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

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

- **Reseller:** reseller who owns this invoice template. Please note, that it doesn't mean that the template will be used for the reseller customers by default. After creation, invoice template still need to be linked to the reseller customers.
- **Name:** unique invoice template name to differentiate invoice templates if there are some.
- **Type:** currently Sipwise C5 supports only svg format of the invoice templates.

All form fields are mandatory.



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

### 7.20.3.2 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 Sipwise C5 provides embedded WYSIWYG SVG editor `svg-edit 2.6` to customize default template. The Sipwise C5 `svg-edit` has some changes in layers management, image edit, user interface, but this [basic introduction](#) still may be useful.

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

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

The screenshot displays the 'Invoice template TestInvoice' editor in the Sipwise NGCP Dashboard. On the left, a 'Template variables' panel lists various variables such as `rescontact.bankname`, `rescontact.bic`, `rescontact.city`, `rescontact.company`, `rescontact.comregnum`, `rescontact.country`, `rescontact.faxnumber`, `rescontact.firstname`, `rescontact.gender`, `rescontact.gpp0`, `rescontact.gpp1`, `rescontact.gpp2`, `rescontact.gpp3`, `rescontact.gpp4`, `rescontact.gpp5`, `rescontact.gpp6`, `rescontact.gpp7`, `rescontact.gpp8`, `rescontact.gpp9`, and `rescontact.iban`. The main editor window shows a preview of an invoice template with embedded variables like `[% custcontact.firstname %] [% custcontact.lastname %]`, `[% custcontact.street %]`, `[% custcontact.postcode %] [% custcontact.city %]`, `[% custcontact.country %]`, `[% invoice.serial %]`, `[% customerexternal_id %]`, `[% p_start %] [% p_end %]`, and `[% date_now(format='%Y-%m-%d') %]`. The preview text includes 'Your Monthly Statement' and 'Dear Customer,'.

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

After implementation of the desired template changes, invoice template should be [saved](#).

To return to Sipwise C5 invoice template **default** content you can press on the "Discard changes" button.



### Important

"Discard changes" operation can't be undone.

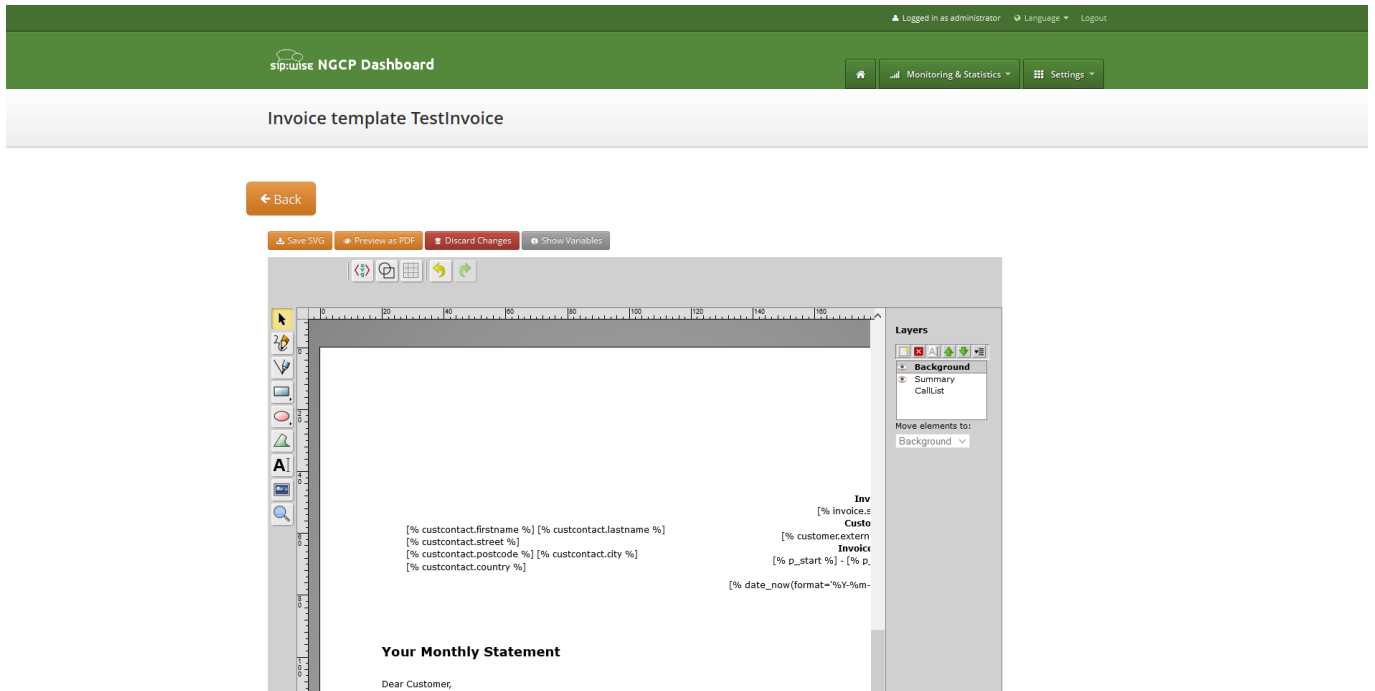
## Layers

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

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

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





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:



Logged in as administrator | Language | Logout

sip:wise NGCP Dashboard | Monitoring & Statistics | Settings

Invoice template Rechnung\_v1

← Back

Save SVG | Preview as PDF | Discard Changes | Show Variables

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

**Your Monthly Statement**  
 Dear Customer,

**Invoice Nr.**  
 [% invoice.serial %]  
**Customer Nr.**  
 [% customer.external\_id %]  
**Invoice Period**  
 [% p\_start %] - [% p\_end %]  
**Date**  
 [% date\_now(format="%Y-%m-%d") %]

SVG XML source of the invoice template will be shown.

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

---

### Note

Template keeps sizes and distances in pixels.

---



### Important

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

---

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

Logged in as administrator | Language | Logout

sip:wise NGCP Dashboard

Monitoring & Statistics | Settings

## Invoice template Default

← Back

2 Save SVG | Preview as PDF | Discard Changes | Show Variables

```

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

```

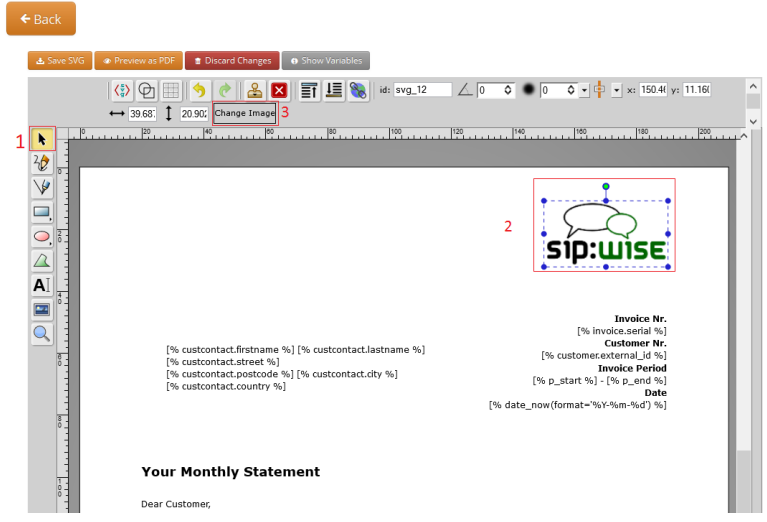
And then [save invoice template changes](#).

### 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

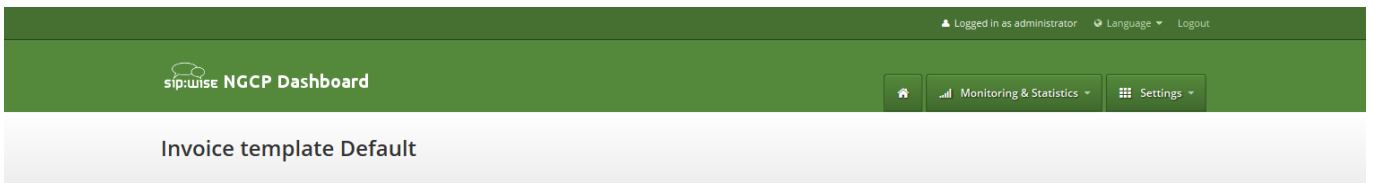
- Make sure that "Select tool" is active.
- Select default logo, clicking on the logo image.
- Press "Change image" button, which should appear on the top toolbar.



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

### 7.20.3.3 Save and preview invoice template content

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



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

Logged in as administrator Language Logout

sip:wise NGCP Dashboard

Monitoring & Statistics Settings

### Invoice template Default

← Back

Invoice template successfully saved



Invoice preview will be opened in the new window.

#### Note

Example fake data will be used for preview generation.

The screenshot shows a browser window displaying a preview of an invoice. The invoice includes the sip:wise logo, customer information, invoice details, and a table of recurring fees.

Customerfirst Customerlast  
Customerstreet 12/3  
12345 Customercity  
Customercountry

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

#### Your Monthly Statement

Dear Customer,

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

| Name                 | Quantity | Unit Price | Total Price in EUR |
|----------------------|----------|------------|--------------------|
| Test Billing Profile | 1        | 29.90      | 29.90              |
| <b>Total</b>         |          |            | <b>29.90</b>       |

## 7.21 Email Reports and Notifications

### 7.21.1 Email events

The Sipwise C5 makes it possible to customize content of the emails sent on the following actions:

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

### 7.21.2 Initial template values and template variables

Default email templates for each of the email events are inserted on the initial Sipwise C5 database creation. Content of the default template is described in the corresponding sections. Default email templates aren't linked to any reseller and can't be changed through Sipwise C5 Panel. They will be used to initialize default templates for the newly created reseller.

Each email template refers to the values from the database using special mark-ups "[%" and "%]". Each email template has fixed set of the variables. Variables can't be added or changed without changes in Sipwise C5 Panel code.

### 7.21.3 Subscriber password reset email template

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

Default content of the password reset email template is:

|                      |   |
|----------------------|---|
| <b>Template name</b> | <b>passreset_default_email</b>  |
| <b>From</b>          | default@sipwise.com   |
| <b>Subject</b>       | Password reset email  |
| <b>Body</b>          | <pre> Dear Customer,  Please go to [%url%] to set your password and log into your self-care ← interface.  Your faithful Sipwise system  --  This is an automatically generated message. Do not reply.</pre> |

Following variables will be provided to the email template:

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

#### 7.21.4 Administrator password reset email template

Email will be sent after administrator requested password reset for it's account. Letter will be sent to the administrator. If the administrator doesn't have an email, an error message will appear when requesting the password reset.

Default content of the password reset email template is:

|                      |   |
|----------------------|---|
| <b>Template name</b> | <code>admin_passreset_default_email</code>  |
| <b>From</b>          | <code>default@sipwise.com</code>  |
| <b>Subject</b>       | Password reset email  |
| <b>Body</b>          | <pre> Dear Customer,  Please go to [%url%] to set your password and log into your admin ← interface.  Your faithful Sipwise system  --  This is an automatically generated message. Do not reply.</pre> |

Following variables will be provided to the email template:

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

#### 7.21.5 New subscriber notification email template

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

---

#### Note

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

---

|                      |                                       |
|----------------------|---------------------------------------|
| <b>Template name</b> | <code>subscriber_default_email</code> |
|----------------------|---------------------------------------|

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

Following variables will be provided to the email template:

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

#### 7.21.6 Invoice email template

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

Variables passed to the email template:

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

---

**Invoice fields**

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

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

---

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



---

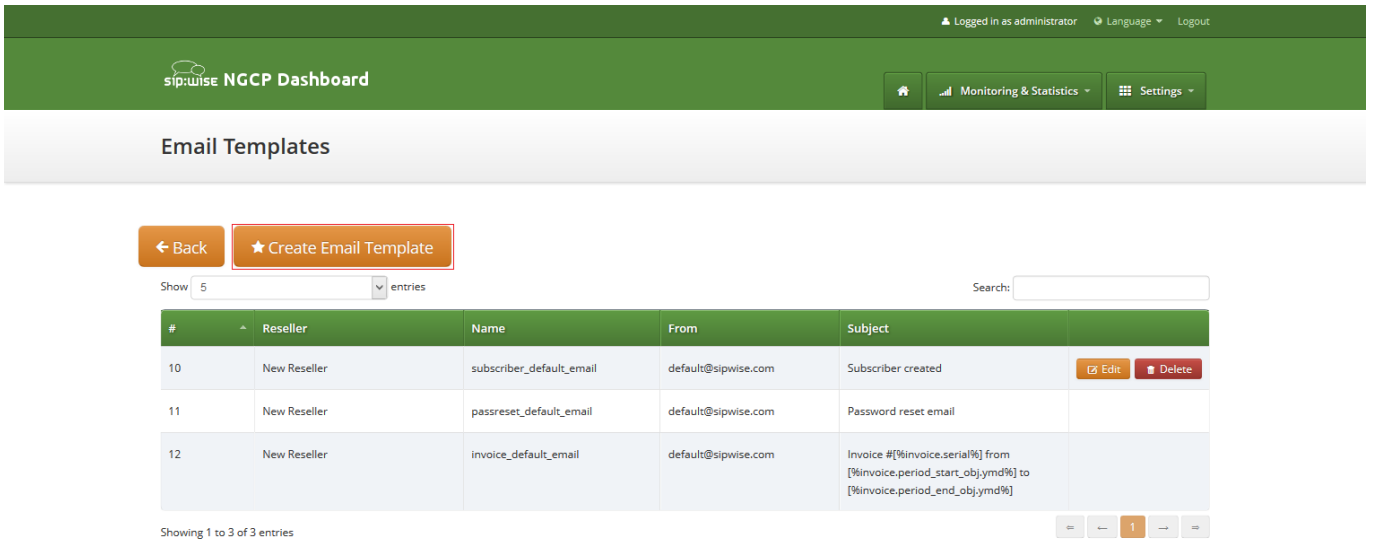
**Contact fields example for the "provider". Replace "provider" to client to access proper "customer" contact fields.**

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

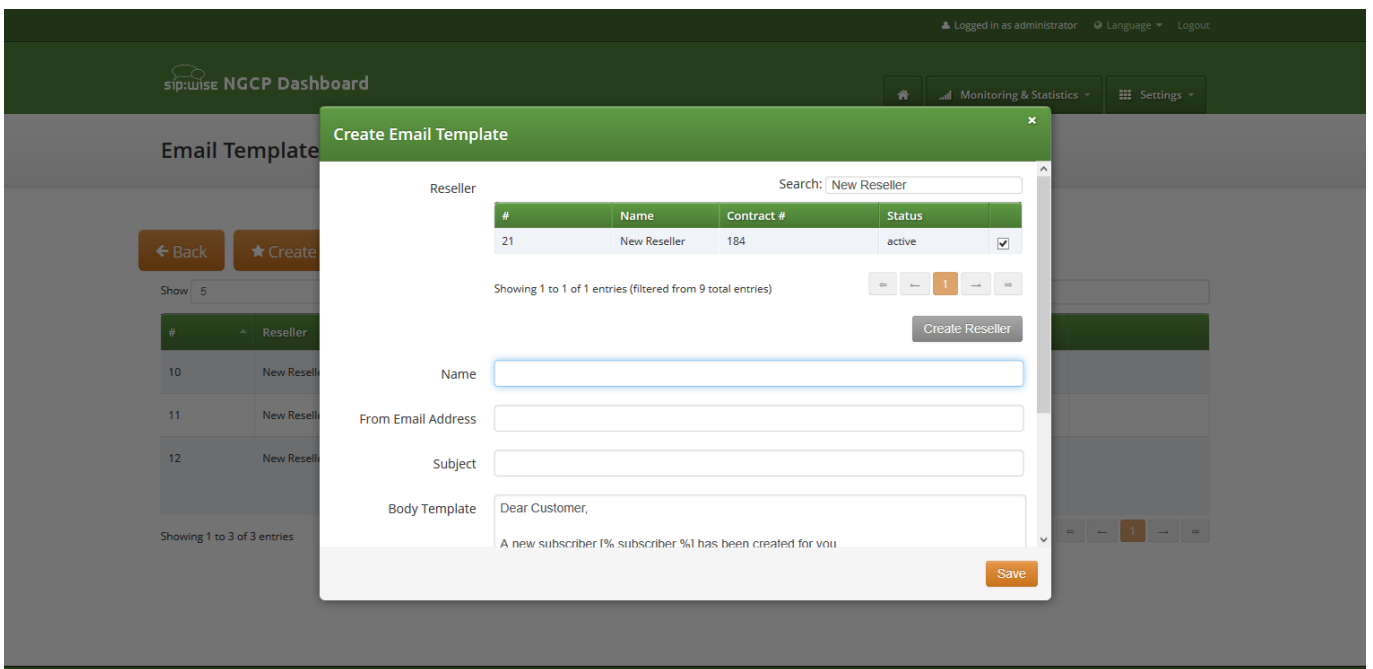
---

### 7.21.7 Email templates management

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



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



On the email template form all fields are mandatory:

- **Reseller:** reseller who owns this email template.
- **Name:** currently only email template with the following names will be considered by Sipwise C5 on the [appropriate event](#) :
  - admin\_passreset\_default\_email;
  - passreset\_default\_email;
  - subscriber\_default\_email;

- invoice\_default\_email;
- **From Email Address:** email address which will be used in the From field in the letter sent by Sipwise C5 .
- **Subject:** Template of the email subject. Subject will be processed with the same template variables as the email body.
- **Body:** Email text template. Will be processed with appropriate template variables.

## 7.22 The Vertical Service Code Interface

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



### Important

The *value* user input is normalized using the Rewrite Rules Sets assigned to domain as described in Section 6.7.

---

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

- **72** - enable *Call Forward Unconditional* e.g. to 431000 by dialing `*72*01000`, and disable it by dialing `#72`.
- **90** - enable *Call Forward on Busy* e.g. to 431000 by dialing `*90*01000`, and disable it by dialing `#90`.
- **92** - enable *Call Forward on Timeout* e.g. after 30 seconds of ringing to 431000 by dialing `*92*30*01000`, and disable it by dialing `#92`.
- **93** - enable *Call Forward on Not Available* e.g. to 431000 by dialing `*93*01000`, and disable it by dialing `#93`.
- **96** - disable at once all the *Call Forwards* previously configured using VSC, e.g. disable all the CFs by dialing `#96`.
- **50** - set *Speed Dial Slot*, e.g. set slot 1 to 431000 by dialing `*50*101000`, which then can be used by dialing `*1`. There is no code to disable a speed dial slot. When a slot is no longer necessary, it can be ultimately removed using the web interface or can be just ignored, because it is not impacting the calls from and to this subscriber.
- **55** - set *One-Shot Reminder Call* e.g. to 08:30 by dialing `*55*0830`.
- **31** - 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.
- **95** - allow to redial the last dialed number by the subscriber. Note: the feature has to be enabled for the subscriber/domain using preference *last\_number\_redial*.

- **71** - allow to hear a voice announcement of the last caller's number, after the announcement dial the key defined in `sems→vsc→callback` to return the call. Note: it is not possible return a call if the caller's number is unavailable.
- **74** - allow to return the call to the last caller's number who called you without hearing the last call ID announcement. Note: it is not possible return a call if the caller's number is unavailable.
- **20** - allow to remove the records of the recent calls to and from you.



### Important

In order to use the feature codes related to recent calls (**95, 71, 74, 20**) the `kamailio→proxy→store_recentcalls` preference in `/etc/ngcp-config/config.yml` has to be set to `yes`.

## 7.22.1 Configuration of Vertical Service Codes

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



### Caution

If you have the EMTAs under your control, make sure that the specified VSCs don't overlap with EMTA-internal VSCs, because the VSC calls must be sent to Sipwise C5 via SIP like normal telephone calls.

## 7.22.2 Voice Prompts for Vertical Service Code Configuration

Table 8: VSC Voice Prompts

| Prompt Handle            | Related VSC                     | Message  |
|--------------------------|---------------------------------|--|
| <code>vsc_error</code>   | any                             | An error has occurred. Please try again later.                     |
| <code>vsc_invalid</code> | wrong code                      | Invalid feature code.  |
| <code>reject_vsc</code>  | any                             | Vertical service codes are disabled for this line.                 |
| <code>vsc_cfu_on</code>  | 72 (Call Forward Unconditional) | Your unconditional call forward has successfully been activated.   |
| <code>vsc_cfu_off</code> | 72 (Call Forward Unconditional) | Your unconditional call forward has successfully been deactivated. |
| <code>vsc_cfb_on</code>  | 90 (Call Forward Busy)          | Your call forward on busy has successfully been activated.         |
| <code>vsc_cfb_off</code> | 90 (Call Forward Busy)          | Your call forward on busy has successfully been deactivated.       |
| <code>vsc_cft_on</code>  | 92 (Call Forward on Timeout)    | Your call forward on ring timeout has successfully been activated. |

Table 8: (continued)

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

## 7.23 Handling WebRTC Clients

WebRTC is an open project providing browsers and mobile applications with Real-Time Communications (RTC) capabilities. Configuring your platform to offer WebRTC is quite easy and straightforward. This allows you to have a SIP-WebRTC bridge in place and make audio/video call towards normal SIP users from WebRTC clients and vice versa. Sipwise C5 listens, by default, on the following WebSockets and WebSocket Secure: `ws://your-ip:5060/ws`, `wss://your-ip:5061/ws` and `wss://your-ip:1443/wss/sip/`.

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

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

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

- Transparent (Pass through using the client's transport protocol)
- RTP/AVP (Plain RTP)
- RTP/SAVP (encrypted SRTP)

- RTP/AVPF (RTP with RTCP feedback)
- RTP/SAVPF (encrypted SRTP with RTCP feedback)
- UDP/TLS/RTP/SAVP (Encrypted SRTP using DTLS)
- UDP/TLS/RTP/SAVPF (Encrypted SRTP using DTLS with RTCP feedback)

**Warning**

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

---

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

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

This will teach Sipwise C5 to translate between Plain RTP and RTP/SAVPF when you have calls between normal SIP clients and WebRTC clients.

## 7.24 XMPP and Instant Messaging

Instant Messaging (IM) based on XMPP comes with Sipwise C5 out of the box. Sipwise C5 uses `prosody` as internal XMPP server. Each subscriber created on the platform have assigned a XMPP user, reachable already - out of the box - by using the same SIP credentials. You can easily open an XMPP client (e.g. Pidgin) and login with your SIP `username@domain` and your SIP `password`. Then, using the XMPP client options, you can create your buddy list by adding your buddies in the format `user@domain`.

## 7.25 Call Recording

### 7.25.1 Introduction to Call Recording Function

Sipwise C5 provides an opportunity to record call media content and store that in files. This function is available since mr5.3.1 version of Sipwise C5 .

**Some characteristics of the Call Recording:**

- Call Recording function can store both unidirectional (originating either from caller, or from callee) or bidirectional (combined) streams from calls, resulting in 1, 2 or 3 physical files as output
- The location and format of the files is configurable.
- File storage is planned to occur on an NFS shared folder.
- Activation of call recording may happen generally for a *Domain / Peer / Subscriber* through Sipwise C5 admin web interface.

**Important**

NGCP's Call Recording function is not meant for individual call interception purpose! Sipwise provides its Lawful Interception solution for that use case.

---

- Querying or deletion of existing recordings may happen through the REST API.
- Listing recordings of a subscriber is possible on NGCP's admin web interface.

The Call Recording function is implemented using NGCP's *rtengine* module.

---

**Note**

There are 2 *rtengine* daemons employed when call recording is enabled and active. The *main rtengine* takes care of forwarding media packets between caller and callee, as usual, while the *secondary rtengine* (recording) daemon is responsible for storing call data streams in the file system.

---

Call Recording is disabled by default. Enabling and configuration of Call Recording takes place in 2 steps:

1. Enabling the feature on Sipwise C5 by setting configuration parameters in the main `config.yml` configuration file.
2. Activating the feature for a *Domain / Peer / Subscriber*.

**7.25.2 Information on Files and Directories**

NGCP's Call Recording function uses an **NFS shared folder** to save recorded streams.

---

**Important**

Since call data amount may be huge (depending, of course, on the number and duration of calls), it is *strongly not recommended* to store recorded streams on NGCP's local disks. However if you *have to* store recorded streams as files in the local filesystem, please contact Sipwise Support team in order to get the proper configuration of Call Recording function.

---

The NFS share gets mounted during startup of the recording daemon. If the NFS share cannot be mounted for some reason, the recording daemon will not start.

**Filenames** have the format: `<call_ID>-<random>-<SSRC>.<extension>`, where:

- `call_ID`: SIP Call-ID of the call being recorded
  - `random`: is a string of random characters, unique for each recorded call. It's purpose is to avoid possible filename collisions if a Call-ID ever gets reused.
  - `SSRC`: is the RTP SSRC for unidirectional recordings, or "mix" for the bidirectional (combined) audio.
  - `extension`: is either "mp3" or "wav", depending on the configuration (`rtproxy.recording.output_format`)
-

There might be 1, 2 or 3 files produced as recorded streams. The **number of files** depends on the configuration:

1. `rtpproxy.recording.output_mixed = 'yes'` (combined stream required)  
`rtpproxy.recording.output_single = 'no'` (unidirectional streams not required)
2. `rtpproxy.recording.output_mixed = 'no'` (combined stream not required)  
`rtpproxy.recording.output_single = 'yes'` (unidirectional streams required)
3. `rtpproxy.recording.output_mixed = 'yes'` (combined stream required)  
`rtpproxy.recording.output_single = 'yes'` (unidirectional streams required)

### 7.25.3 Configuration

The Call Recording function can be enabled and configured on Sipwise C5 by changing the following configuration parameters in `config.yml` file:

```
rtpproxy:
  ...
  recording:
    enable: no
    mp3_bitrate: '48000'
    nfs_host: 192.168.1.1
    nfs_remote_path: /var/recordings
    output_dir: /var/lib/rtpengine-recording
    output_format: wav
    output_mixed: yes
    output_single: yes
    resample: no
    resample_to: '16000'
    spool_dir: /var/spool/rtpengine
```

#### 7.25.3.1 Enabling Call Recording

Enabling the function requires changing the value of `rtpproxy.recording.enable` parameter to "yes". In order to make the new configuration active, it's necessary to do:

```
ngcpcfg apply 'Activated call recording'
```

#### Description of configuration parameters:

- `enable`: when set to "yes" Call Recording function is enabled; default: "no"
- `mp3_bitrate`: the bitrate used when recording happens in MP3 format; default: "48000"



- `nfs_host`: IP address of the NFS host that provides storage space for recorded streams; default: "192.168.1.1"
- `nfs_remote_path`: the remote path (folder) where files of recorded streams are stored on the NFS share; default: "/var/recordings"
- `output_dir`: is the local mount point for the NFS share, and thus where the final audio files will be written; default: "/var/lib/rtpengine-recording"

**Caution**

Normally you don't need to change the default setting. If you do change the value, please be aware that recorded files will be written by `root` user in that directory.

- `output_format`: possible values are "wav" (Wave) or "mp3" (MP3); default: "wav"
- `output_mixed`: "yes" means that there is a file that contains a mixed stream of caller and callee voice data; default: "yes"
- `output_single`: "yes" means that there is a separate file for each stream direction, i.e. for the streams originating from caller and callee; default: "yes"
- `resample`: when set to "yes" the call data stream will be resampled before storing it in the file; default: "no"
- `resample_to`: the sample rate used for resampling output; default: "16000"
- `spool_dir`: is the place for temporary metadata files that are used by the recording daemon and the main `rtpengine` daemon for their communication; default: "/var/spool/rtpengine"

**Caution**

You should not change the default setting unless you have a good reason to do so! Sipwise has thoroughly tested the Call Recording function with the default setting.

If Call Recording is enabled you can see 2 `rtpengine` processes running when checking Sipwise C5 system state with `ngcp-service` tool:

```
root@sp1:/etc/ngcp-config# ngcp-service summary
Ok Service                               Managed   Started   Status
-- -----
...
  kamailio-lb                             managed   by-ha     active
  ngcp-voisniff                           managed   by-ha     active
  rtpengine                                managed   by-ha     active
  rtpengine-recording                      managed   by-ha     active
...
```

### 7.25.3.2 Activating Call Recording

Activating Call Recording for e.g. a *Subscriber*: please use NGCP's admin web interface for this purpose. On the web interface one has to navigate as follows: *Settings* → *Subscribers* → *select subscriber Details* → *Preferences* → *NAT and Media Flow Control*. Afterwards the `record_call` option has to be enabled by pressing the *Edit* button and ticking the checkbox.

| NAT and Media Flow Control |                    |  |   |                                     |
|----------------------------|--------------------|--|---|-------------------------------------|
|                            | Attribute          | Name                                   | Value   |                                     |
|                            | sound_set          | System Sound Set                       | <input type="text" value=""/>                   |                                     |
|                            | use_rtpproxy       | RTP-Proxy Mode                         | <input type="text" value="use domain default"/> |                                     |
|                            | ipv46_for_rtpproxy | IPv4/IPv6 bridging mode                | <input type="text" value="use domain default"/> |                                     |
|                            | contract_sound_set | Customer Sound Set                     | <input type="text" value=""/>                   |                                     |
|                            | music_on_hold      | Music on Hold                          | <input type="checkbox"/>                        |                                     |
|                            | bypass_rtpproxy    | Disable RTP-Proxy in the selected case | <input type="text" value="use domain default"/> |                                     |
|                            | rtp_interface      | RTP interface                          | <input type="text" value="default"/>            |                                     |
|                            | transport_protocol | Media transport protocol               | <input type="text" value="use domain default"/> |                                     |
|                            | set_moh_sendonly   | MoH sendonly                           | <input type="checkbox"/>                        |                                     |
|                            | codecs_filter      | Codecs filter                          | <input type="checkbox"/>                        |                                     |
|                            | codecs_list        | Codecs list                            |   |                                     |
|                            | record_call        | Record calls                           | <input type="checkbox"/>                        | <input type="button" value="Edit"/> |

Figure 60: Activating Call Recording

#### Note

The call recording function may be activated for a single *Subscriber*, a *Domain* and a *Peer* server in the same way: *Preferences* → *NAT and Media Flow Control* → *record\_call*. When activating call recording for a *Domain* or *Peer* this effectively activates the function for all subscribers that belong to the selected domain, and for all calls with a local endpoint going through the selected peer server, respectively.

It is possible to **list existing call recordings** of a *Subscriber* through the admin web interface of NGCP. In order to do so, please navigate to: *Settings* → *Subscribers* → *select subscriber Details* → *Call Recordings*

Subscriber 43993002@10.15.18.222

← Back Preferences Calls history Customer Expand Groups

Master Data  
Groups  
Voice Mails  
Call Recordings

From Date:  To Date:  Search:

| # | Time                    | Call-ID                              |
|---|-------------------------|--------------------------------------|
| 1 | 2017-09-05 11:51:41.543 | 17d5b961-7613-4ba2-9bc4-fb8086e2ccdd |

Showing 1 to 1 of 1 entries

Figure 61: Listing Call Recordings

If you select an item in the list, besides the main properties such as the time of call and the SIP Call-ID, you can retrieve the details of the related call (press the *Call Details* button), get the list of recorded files (press the *Recorded Files* button) or *Delete* the recorded call.

When selecting *Call Details* you will see the most important accounting data of the call. Furthermore you can see the SIP *Call Flow* or the complete *Call Details* if you press the respective buttons.

Call List for 43993002@10.15.18.222 ( )

← Back

Show all calls

Show 5 entries From Date:  To Date:  Search:

| #     | Caller   | Callee   | CLIR | Billing zone     | Status | Start Time              | Duration    | Call-ID                              | Cost |
|-------|----------|----------|------|------------------|--------|-------------------------|-------------|--------------------------------------|------|
| 5     | 43993002 | 43993003 | 0    | All Destinations | ok     | 2017-09-05 11:51:47.855 | 0:00:10.437 | 17d5b961-7613-4ba2-9bc4-fb8086e2ccdd | 0.00 |
| Total |          |          |      |                  |        |                         | 0:00:10.437 |                                      | 0.00 |

Showing 1 to 1 of 1 entries

Figure 62: Listing Call Details for a Recording

When navigating to *Recorded Files* of a call you will be presented with a list of files. For each file item:

- type of stream is shown, that can be either "mixed" (combined voice data), or "single" (voice data of caller or callee)

- file `format` is shown, that can be either "wav", or "mp3"
- you can download the file by pressing the *Play* button

| # | Type   | Format |
|---|--------|--------|
| 1 | mixed  | wav    |
| 3 | single | wav    |
| 5 | single | wav    |

Showing 1 to 3 of 3 entries

Figure 63: Listing Files for a Recording

#### 7.25.4 REST API

The Sipwise C5 REST API provides methods for querying and deletion of existing recording data. The full documentation of the available API methods is available on the admin web interface of the NGCP, as usual.

The following API methods are provided for managing Call Recordings:

- CallRecordings:
  - Provides information about the calls recorded in the system; can also be used to delete a recording entry
  - accessible by the path: `/api/callrecordings` (collection) or `/api/callrecordings/id` (single item)
  - Supported HTTP methods: OPTIONS, GET, DELETE
- CallRecordingStreams:
  - Provides information about recorded streams, such as start time, end time, format, mixed/single type, etc.; can also be used to delete a recorded stream
  - accessible by the path: `/api/callrecordingstreams` (collection) or `/api/callrecordingstreams/id` (single item)
  - Supported HTTP methods: OPTIONS, GET, DELETE
- CallRecordingFiles:
  - Provides information about recorded streams, such as start time, end time, format, mixed/single type, etc.; additionally returns the file content too
  - accessible by the path: `/api/callrecordingfiles` (collection) or `/api/callrecordingfiles/id` (single item)
  - Supported HTTP methods: OPTIONS, GET

### 7.25.5 Pre-Recording Announcement

Many country regulations require that an informative announcement is played to the caller before the call is actually recorded. The Sipwise C5 allows you to configure your own custom announcement with few simple steps.

First create a system sound set for the feature. In *Settings* → *Sound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is an announcement *early\_rejects* → *announce\_before\_recording* for that purpose.

Once the *Sound Set* is created the subscriber's preference *play\_announce\_before\_recording* of the callee must be enabled under *Subscriber* → *Preferences* → *Applications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers.

---

#### Note

The announcement will be played to caller before the call is routed to the callee.

---



#### Important

In case of **CFU** or **CFNA** with Pre-Recording Announcement feature enabled on both the forwarder and the final callee, only the Announcement of final callee will be played to caller. In case of **CFB** and **CFT**, instead, the announcement of the forwarder will be played first, then the announcement of final callee will be played after the call forward is executed.

---

## 7.26 Media Transcoding

### 7.26.1 Overview

Starting with version mr6.2.1, Sipwise C5 offers the capability to convert RTP media between several supported codecs, a feature known as transcoding. While this feature is always available on Sipwise C5, it's engaged only when a subscriber, peer, or domain is explicitly configured for it. By default, Sipwise C5 lets RTP endpoints negotiate the codec to use among themselves without interfering.



#### Important

Media transcoding is a relatively CPU-intensive feature. As such, each individual node of a Sipwise C5 performing media transcoding can only support a limited number of concurrent calls for which transcoding is active.

---

### 7.26.2 Supported Codecs

The following audio codecs, which are commonly found in use by SIP/RTP clients, are currently supported for transcoding.

- G.711 ( $\mu$ -Law and a-Law)
- G.722

- G.723.1
- G.729
- GSM
- AMR (narrowband and wideband, the latter also known as AMR-WB)
- Opus
- Speex
- DTMF event packets (**telephone-event**)

Some codecs operate at different sampling rates than other codecs. If transcoding happens between two such codecs, the audio will be resampled as necessary. Similarly, if transcoding happens between a mono (1-channel) and a stereo (2-channel) codec, the audio will be up-mixed and down-mixed as necessary.

### 7.26.3 Configuration

Transcoding can be engaged for individual subscribers, peers, or domains on their respective preferences page in the Sipwise C5 admin web interface.

| Media Codec Transcoding Options     |                       |                                      |                          |
|-------------------------------------|-----------------------|--------------------------------------|--------------------------|
| Attribute                           | Name                  | Value                                |                          |
| <input checked="" type="checkbox"/> | ptime                 | RTP packet interval                  | use domain default ▾     |
| <input checked="" type="checkbox"/> | transcode_PCMU        | Transcode to G.711 μ-Law             | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_PCMA        | Transcode to G.711 a-Law             | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_G722        | Transcode to G.722                   | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_G723        | Transcode to G.723.1                 | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_G729        | Transcode to G.729                   | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_GSM         | Transcode to GSM                     | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_AMR         | Transcode to AMR                     | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_AMR_WB      | Transcode to AMR-WB                  | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_opus_mono   | Transcode to Opus mono               | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_opus_stereo | Transcode to Opus stereo             | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_speex_8     | Transcode to Speex 8 kHz             | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_speex_16    | Transcode to Speex 16 kHz            | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | transcode_speex_32    | Transcode to Speex 32 kHz            | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | opus_mono_bitrate     | Opus mono bitrate                    | use domain default ▾     |
| <input checked="" type="checkbox"/> | opus_stereo_bitrate   | Opus stereo bitrate                  | use domain default ▾     |
| <input checked="" type="checkbox"/> | g723_bitrate          | G.723.1 bitrate                      | use domain default ▾     |
| <input checked="" type="checkbox"/> | always_transcode      | Always transcode media from the user | <input type="checkbox"/> |

Figure 64: Transcoding Configuration

Setting any of the transcoding options for a domain makes it affect all the subscribers in this domain.

Individual options are described below.

### 7.26.3.1 ptime

Packetisation time in milliseconds. Normally Sipwise C5 lets the RTP endpoints select and negotiate the packetisation time they want to use. Setting this option to anything other than `unchanged` will engage the transcoding engine towards this subscriber or peer even if none of the other transcoding options are set, in which case the media will simply be repacketised.

For example, setting this to `40 ms` would mean that each RTP packet sent towards this subscriber or peer would contain 40 milliseconds worth of audio, even if the other side of the call sends media that is packetised differently. It would also make Sipwise C5 indicate towards this subscriber or peer that it would prefer to receive audio in 40 millisecond packets (through the `a=ptime` SDP attribute).

### 7.26.3.2 transcode\_...

Enabling one of these options adds the selected codecs to the list of codecs offered to this subscriber or peer, even if the original list of offered codecs did not include it. If this additional codec ends up being accepted by this subscriber or peer, then it will be transcoding to the first supported codec that was originally offered.

For example, if a calling RTP client A indicates support for PCMA (G.711 a-Law) as well as G.722, and calls a subscriber B that is configured for transcoding to G.729, then subscriber B would be offered PCMA, G.722, and G.729 by Sipwise C5. If subscriber B then accepts G.729 and starts sending G.729, Sipwise C5 would engage its transcoding engine and transcode the audio to PCMA (because PCMA and not G.722 was the codec preferred by A) before forwarding it to A. Vice versa, PCMA arriving from A would be transcoded to G.729 before being sent to B. (If B were to reject G.729 and instead starts to send PCMA or G.722, no transcoding would happen.)

Notes on individual codecs:

- **AMR** is available in both narrowband (`AMR` operating at 8 kHz) and wideband (`AMR-WB` operating at 16 kHz) variants. These are distinct codecs and can be configured for transcoding separately or together.
- **Opus** always operates at 48 kHz, but is supported in both mono and stereo (1 and 2 audio channels respectively). Both can be offered at the same time if so desired.
- **Speex** is supported at sampling rates of 8, 16, and 32 kHz. These can be configured separately for transcoding, or together.
- **DTMF** is not an actual audio codec, but rather represents transcoding between DTMF event packets and in-band DTMF audio tones. This is described in more detail below.

### 7.26.3.3 ...\_bitrate

Some codecs (Opus and G.723.1 in particular) can be configured for different bitrates, which would impact the amount of network bandwidth they use, as well as the audio quality produced. For Opus, different bitrates can be selected for their mono and stereo instances. Selecting a bitrate has no effect if transcoding to the respective codec is not engaged.

#### 7.26.3.4 `always_transcode`

Setting this flag instructs Sipwise C5 to always engage transcoding to the first (preferred) codec indicated by an RTP endpoint, even if another codec is available that is supported by both parties to a call. Enabling this flag can potentially engage the transcoding engine for a call even if none of the other transcoding options are set.

For example: Subscriber A is calling subscriber B. Subscriber A is indicating support for PCMA and G.722. Subscriber B answers the call, rejects PCMA but accepts G.722, and starts sending G.722 to A. Normally Sipwise C5 would not get involved and would simply let G.722 pass between A and B. But if subscriber B has the `always_transcode` flag set, Sipwise C5 would now start transcoding the G.722 sent by B into PCMA before forwarding it to A, because PCMA was indicated as the preferred codec by A. Vice versa, PCMA arriving from A would be transcoded into G.722 and then forwarded to B.

#### 7.26.3.5 `DTMF transcoding`

Sipwise C5 supports transcoding between DTMF event packets (using the RTP `telephone-event` type payload) and DTMF tones carried in-band in the audio stream. DTMF transcoding is supported in both directions: transcoding DTMF event packets to DTMF tones, and DTMF tones in an audio stream and transcoding them to DTMF event packets.

Support for DTMF transcoding can be enabled in one of two ways:

- Enabling the setting `transcode_dtmf` for a subscriber, peer, or domain. This is useful if the subscriber, peer, or domain requires support for DTMF event packets, but the calling entity might only support DTMF tones carried in-band in the audio stream.
- Enabling the setting `always_transcode` for a subscriber, peer, or domain. This is useful for the reverse case: if the subscriber, peer, or domain might only support DTMF tones carried in-band in the audio stream, but the calling entity requires support for DTMF event packets.

Enabling DTMF transcoding for any call requires that all audio passes through the transcoding engine, as well as a DSP for detecting DTMF tones in one direction. This carries an additional performance impact with it, and so DTMF transcoding should only be enabled when really necessary.

#### 7.26.3.6 `DTMF conversion of INFO messages`

Sipwise C5 supports the conversion of SIP INFO messages with `application/dtmf-relay` payload to DTMF event or PCM DTMF inband tone, depending on whether the destination participant supports the `telephone-event` RTP payload type or not. DTMF conversion of INFO messages works only in one way direction: DTMF events or PCM DTMF inband tones are not converted back to DTMF INFO messages.

The preference `kamailio.proxy.allow_info_method` has to be set to `yes` in `config.yml` in order to make the DTMF INFO message conversion working.

Enabling DTMF conversion of INFO messages for any call requires that all audio passes through the transcoding engine, as well as a DSP for detecting DTMF tones in one direction. This carries an additional performance impact with it, and so it should only be enabled when really necessary.



---

**Note**

At the moment this feature is for internal use only. External generated INFO messages will be not converted but passed through.

---

#### 7.26.4 T.38 transcoding

In addition to transcoding between audio codecs, Sipwise C5 supports transcoding between T.38 fax transmissions (over UDPTL transport) and T.30 fax data over regular audio channels. The audio codec commonly used to carry T.30 data is G.711, but any other audio codec that is supported for transcoding can also be used, provided it offers a high enough bitrate and audio quality.

Two settings to control T.38 transcoding are available for subscribers, peers, and domains:

- The setting `t38_decode` instructs Sipwise C5 to accept an offered T.38 session towards the subscriber, peer, or domain, and translate it into a regular audio call carrying T.30 fax data. By default, G.711 (both  $\mu$ -Law and a-Law) will be offered. If any other codecs are selected through the `transcode_...` options, then only those codecs will be offered and the G.711 default will be omitted. Non-T.38 offers are not affected by this settings.
- The setting `t38_force` forces any audio call towards this subscriber, peer, or domain to be translated to a T.38 session, regardless of whether the audio media actually carries T.30 fax data or not. This is useful if it is known that the remote destination is a fax endpoint supporting T.38.

### 7.27 Announcement Before Call Setup

This feature allows a callee to play a custom announcement to the caller every time it receives a call. The announcement is played in early media mode, therefore it can be used as a simple business welcome message or to inform the caller about a different cost of the call before it will be actually charged.

The configuration of the announcement is similar to the activation of [Pre-Recording Announcement](#) and it requires few simple steps.

First create a system sound set for the feature. In *Settings* → *Sound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is an announcement `early_rejects` → `announce_before_call_setup` for that purpose.

Once the *Sound Set* is created the subscriber's preference `play_announce_before_call_setup` must be enabled under *Subscriber* → *Preferences* → *Applications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers.

---

**Note**

The announcement will be played to caller before the call is routed to the callee.

---

**Important**

Differently from *Pre-Recording Announcement*, in all **Call Forward** cases with *Announcement Before Call Setup* feature enabled on both the forwarder and the final callee, only the announcement of the forwarder will be played to caller.

---

This feature and *Pre-Recording Announcement* can be activated at the same time. In this case the *Announcement Before Call Setup* will be played as first.

## 7.28 Store Recent Calls and Redial

Sipwise C5 allows to store the number of the last incoming and outgoing calls of each subscriber. To enable the feature edit `config.yml` and enable there `kamailio: store_recentcalls: yes`. Only the very last incoming and outgoing call is stored.

Each subscriber can interact with his own personal stored records using Vertical Service Codes (see [VSC](#)). In particular a subscriber can:

- redial last dialed number (this feature has to be enabled for the each subscriber/domain using preference `last_number_redial`)
- hear a voice announcement of the last caller's number, then press the key defined in `sems→vsc→callback_last_caller_confirmation_key` preference of `/etc/ngcp-config/config.yml` to return the call
- return the call to the last caller's number without hearing the number announcement
- delete the personal stored records of any recent calls to and from him

---

### Note

it is not possible return a call if the caller's number is unavailable (e.g. anonymous calls).

---

### 7.28.1 Configuring Recent Calls Sound Sets

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

| recent_calls             |          |                          |                        |
|--------------------------|----------|--------------------------|------------------------|
| Name                     | Filename | Loop                     |                        |
| recent_call_anonymous    |          | <input type="checkbox"/> | <a href="#">Upload</a> |
| recent_call_confirmation |          | <input type="checkbox"/> |                        |
| recent_call_deleted      |          | <input type="checkbox"/> |                        |
| recent_call_empty        |          | <input type="checkbox"/> |                        |
| recent_call_play_number  |          | <input type="checkbox"/> |                        |

Upload the following files:

Table 9: Recent Calls Sound Sets

| Handle                   | Message played   |
|--------------------------|--|
| recent_call_play_number  | The last call you received was from . . . .  |
| recent_call_confirmation | To call back this number press key . . . .   |
| recent_call_anonymous    | The last caller didn't share his number, you can not return the call to this person. |
| recent_call_empty        | Your recent call history is empty.   |
| recent_call_deleted      | Your recent call history has been successfully deleted.                              |

**Note**

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

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

**7.28.2 Advanced configuration**

By default the expiration time for the most recent incoming and outgoing call per subscriber are 3600 seconds (1 hour) and 86400 seconds (1 day) respectively. If you wish to prolong or shorten the expiration time open `constants.yml` and set there `recentcalls: expire: 3600` and `recentcalls: out_expire: 86400` to a new value, then issue `ngcpcfg apply "recentcalls expire modification"` afterwards.

**7.29 SMS (Short Message Service) on Sipwise C5**

Starting with its mr5.0.1 release, Sipwise C5 offers *short messaging service* to its local subscribers. The implementation is based on a widely used software module: *Kannel*, and it needs to interact with a mobile operator's SMSC in order to send and receive SMS for the local subscribers. The data exchange with SMSC uses *SMPP* (Short Message Peer-to-Peer) protocol.

**SMS directions:**

- incoming / received: the destination of the SM is a local subscriber on the NGCP
- outgoing / sent: the SM is submitted by a local subscriber

**Note**

The Sipwise C5 behaves as a short message client towards the SMSC of a mobile operator. This means every outgoing SM will be forwarded to the SMSC, and every incoming SM will reach Sipwise C5 through an SMSC.

The architecture of the SMS components of Sipwise C5 and their interaction with other elements is depicted below:

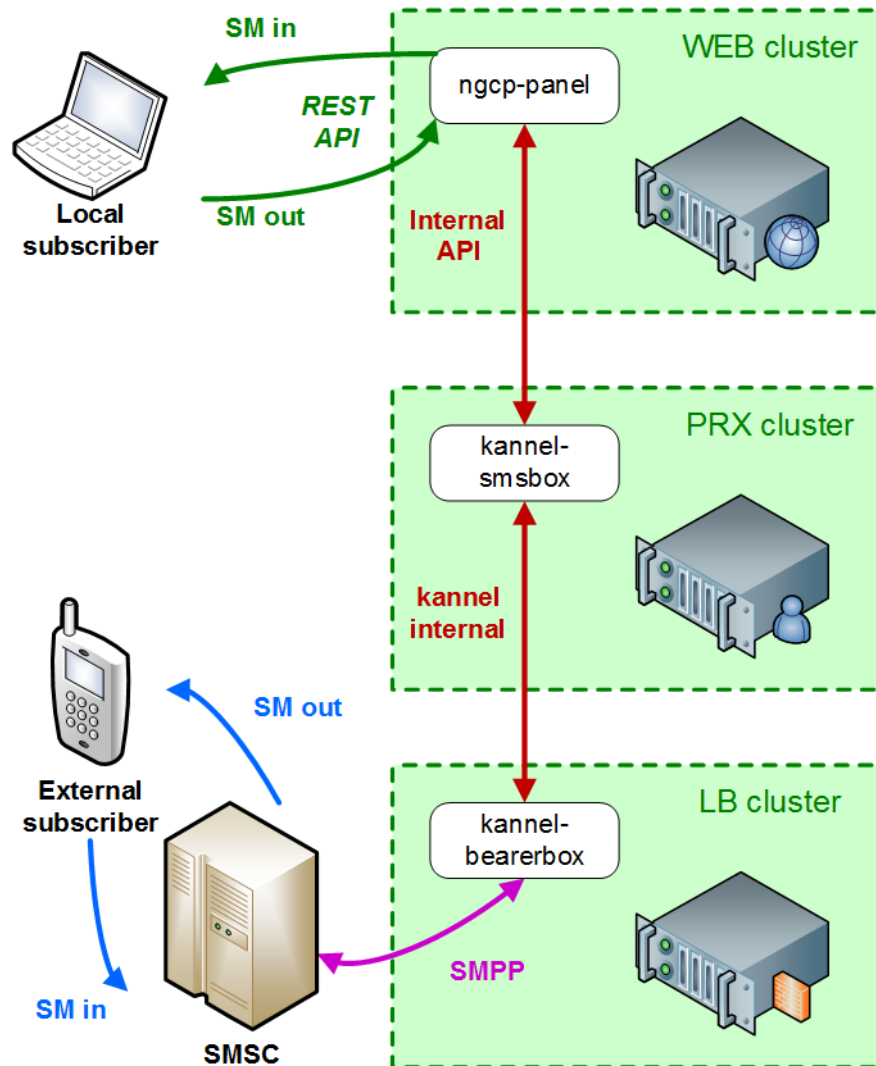


Figure 65: SMS Interaction

**Note**

For the *Sipwise C5 CE and PRO* installations: the *Kannel* components and the *ngcp-panel* all run on the same single node. The description of SMS module will continue referring to a *Sipwise C5 CARRIER* installation in the handbook.

There are 2 components of the SMS module:

- **SMS Box:** this component takes care of handling the messages locally, that means:
  - delivering them to subscribers (writing into database for later retrieval)
  - picking up the submitted SMSs from the database and forwarding them to the *Bearer Box* component
- **Bearer Box:** this component manages the transmission of SMSs between Sipwise C5 and the mobile operator's SMSC

## 7.29.1 Configuration

### 7.29.1.1 Main Parameters

The SMS functionality of Sipwise C5 is disabled by default. In order to **enable SMS**, change the value of configuration parameter `sms.enable` to `yes` in the main configuration file (`/etc/ngcp-config/config.yml`).

The second step of configuration is related to the **SMSC** where Sipwise C5 will connect to. Set the following parameters:

- `sms.smsc.host`: IP address of the SMSC
- `sms.smsc.port`: Port number of the SMSC
- `sms.smsc.username`: Username for authentication on the SMSC
- `sms.smsc.password`: Password for authentication on the SMSC

Other parameters of the SMSC connection may also need to be changed from the default values, but this is specific to each deployment.

Then, as usual, you have to make the new configuration active:

```
$ ngcpcfg apply 'Enabled SMS'
```

### 7.29.1.2 Configuration Files of Kannel

There are a few configuration files for the *Kannel* module, namely:

- `/etc/default/ngcp-kannel`: determines which components of *Kannel* will be started. This is auto-generated from `/etc/ngcp-config/templates/etc/default/ngcp-kannel.tt2` file when SMS is enabled.
- `/etc/kannel/kannel.conf`: contains detailed configuration of *Kannel* components. This is auto-generated from `/etc/ngcp-config/templates/etc/kannel/kannel.conf.tt2` file when SMS is enabled.
- `/etc/logrotate.d/ngcp-kannel.conf`: configuration of *logrotate* for *Kannel* log files. This is auto-generated from `/etc/ngcp-config/templates/etc/logrotate.d/ngcp-kannel.conf.tt2` file when SMS is enabled.



#### Caution

Please do not change settings in the above mentioned template files, unless you have to tailor *Kannel* settings to your specific needs!

---

Finally: see the description of each configuration parameter in the [appendix](#).

### 7.29.1.3 Call Forwarding for SMS (CFS)

Any subscriber registered on Sipwise C5 can apply a call forwarding setting for short messages, referred to as "CFS" (Call Forward - SMS). If the CFS feature is enabled, he can receive the SMs on his mobile phone, for example, instead of retrieving the SMs through the REST API. This is much more convenient for users if they do not have an application on their smartphone or computer that could manage the SMs through the REST API.

In order to enable CFS you have to set the forwarding as usual on the admin web interface, or through the REST API. Navigate to *Subscribers* → *select one* → *Details* → *Preferences* → *Call Forwards* and press the *Edit* button.

Subscriber Preferences for 43993003@10.15.18.222

← Back Expand Groups

Successfully saved Call Forward

Call Forwards

| Type                       | Answer Timeout | Destinations              | Timeset  | Sources |   |
|----------------------------|----------------|---------------------------|----------|---------|---|
| Call Forward Unconditional |                |                           |          |         |   |
| Call Forward Busy          |                |                           |          |         |   |
| Call Forward Timeout       |                |                           |          |         |   |
| Call Forward Unavailable   |                |                           |          |         |   |
| Call Forward SMS           |                | 435551234101@10.15.18.222 | for 300s | always  | all sources <span>Edit</span> <span>Delete</span> |

Figure 66: Call Forward for SMS

## 7.29.2 Monitoring, troubleshooting

### 7.29.2.1 Bearer Box (LB node of NGCP)

On the LB node you can see a **process** named "**bearerbox**". This process has 2 **listening ports** assigned to it:

- 13000: this is the generic *Kannel* administration port, that belongs to the "core" component of Kannel.
- 13001: this is the communication port towards the *SMS Box* component running on PRX nodes of NGCP.

The *ngcp-service* tool also shows the *bearerbox* process in its summary information:

```
$ ngcp-service summary
Ok Service                Managed   Started   Status
-----
```

```
...
kannel-bearerbox          managed    by-ha     active
...
```

The following log files can provide information about the operation of *Bearer Box*:

- status messages and high level, short entries about sent and received messages: `/var/log/ngcp/kannel/kannel.log`

```
...
2017-09-26 08:57:32 [15922] [10] DEBUG: boxc_receiver: heartbeat with load value 0 ↔
    received
...
2017-09-26 11:12:06 [15922] [10] DEBUG: boxc_receiver: sms received
2017-09-26 11:12:06 [15922] [10] DEBUG: send_msg: sending msg to box: <192.168.1.4>
2017-09-26 11:12:06 [15922] [11] DEBUG: send_msg: sending msg to box: <192.168.1.4>
2017-09-26 11:12:06 [15922] [11] DEBUG: boxc_sender: sent message to <192.168.1.4>
2017-09-26 11:12:06 [15922] [10] DEBUG: boxc_receiver: got ack
...
```

- detailed information and message content of sent and received messages, link enquiries: `/var/log/kannel/smsc.log`

---

#### Note

Sent and received message examples shown here do not contain the full phone number and content for confidentiality reason.

---

- Example received message:

```
...
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP[default_smsc]: Got PDU:
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP PDU 0x7f2274025070 dump:
2017-09-26 12:09:36 [15922] [6] DEBUG:   type_name: deliver_sm
2017-09-26 12:09:36 [15922] [6] DEBUG:   command_id: 5 = 0x00000005
2017-09-26 12:09:36 [15922] [6] DEBUG:   command_status: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   sequence_number: 11867393 = 0x00b51501
2017-09-26 12:09:36 [15922] [6] DEBUG:   service_type: NULL
2017-09-26 12:09:36 [15922] [6] DEBUG:   source_addr_ton: 2 = 0x00000002
2017-09-26 12:09:36 [15922] [6] DEBUG:   source_addr_npi: 1 = 0x00000001
2017-09-26 12:09:36 [15922] [6] DEBUG:   source_addr: "0660....."
2017-09-26 12:09:36 [15922] [6] DEBUG:   dest_addr_ton: 1 = 0x00000001
2017-09-26 12:09:36 [15922] [6] DEBUG:   dest_addr_npi: 1 = 0x00000001
2017-09-26 12:09:36 [15922] [6] DEBUG:   destination_addr: "43668....."
2017-09-26 12:09:36 [15922] [6] DEBUG:   esm_class: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   protocol_id: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   priority_flag: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   schedule_delivery_time: NULL
```

```

2017-09-26 12:09:36 [15922] [6] DEBUG:   validity_period: NULL
2017-09-26 12:09:36 [15922] [6] DEBUG:   registered_delivery: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   replace_if_present_flag: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   data_coding: 3 = 0x00000003
2017-09-26 12:09:36 [15922] [6] DEBUG:   sm_default_msg_id: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   sm_length: 158 = 0x0000009e
2017-09-26 12:09:36 [15922] [6] DEBUG:   short_message:
2017-09-26 12:09:36 [15922] [6] DEBUG:     Octet string at 0x7f227400f80:
2017-09-26 12:09:36 [15922] [6] DEBUG:       len: 158
2017-09-26 12:09:36 [15922] [6] DEBUG:       size: 159
2017-09-26 12:09:36 [15922] [6] DEBUG:       immutable: 0
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 5a <14 bytes> 46
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 72 <14 bytes> 68
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 61 <14 bytes> 67
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 20 <14 bytes> 57
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 65 <14 bytes> 63
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 68 <14 bytes> 73
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 2e <14 bytes> 61
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 6c <14 bytes> 73
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 3a <14 bytes> 73
2017-09-26 12:09:36 [15922] [6] DEBUG:       data: 4d <14 bytes> 6e
2017-09-26 12:09:36 [15922] [6] DEBUG:     Octet string dump ends.
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP[default_smsc]: Sending PDU:
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP PDU 0x7f2274020790 dump:
2017-09-26 12:09:36 [15922] [6] DEBUG:   type_name: deliver_sm_resp
2017-09-26 12:09:36 [15922] [6] DEBUG:   command_id: 2147483653 = 0x80000005
2017-09-26 12:09:36 [15922] [6] DEBUG:   command_status: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   sequence_number: 11867393 = 0x00b51501
2017-09-26 12:09:36 [15922] [6] DEBUG:   message_id: NULL
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP[default_smsc]: throughput (0.00,5.00)
...

```

– Example sent message:

```

...
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]: throughput (0.00,5.00)
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]: Manually forced source addr ←
   ton = 1, source add npci = 1
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]: Manually forced dest addr ton ←
   = 1, dest add npci = 1
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]: Sending PDU:
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP PDU 0x7f2274025070 dump:
2017-09-26 12:04:08 [15922] [6] DEBUG:   type_name: submit_sm
2017-09-26 12:04:08 [15922] [6] DEBUG:   command_id: 4 = 0x00000004
2017-09-26 12:04:08 [15922] [6] DEBUG:   command_status: 0 = 0x00000000

```



```

2017-09-26 12:04:08 [15922] [6] DEBUG: sequence_number: 98163 = 0x00017f73
2017-09-26 12:04:08 [15922] [6] DEBUG: service_type: NULL
2017-09-26 12:04:08 [15922] [6] DEBUG: source_addr_ton: 5 = 0x00000005
2017-09-26 12:04:08 [15922] [6] DEBUG: source_addr_npi: 0 = 0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG: source_addr: "any"
2017-09-26 12:04:08 [15922] [6] DEBUG: dest_addr_ton: 1 = 0x00000001
2017-09-26 12:04:08 [15922] [6] DEBUG: dest_addr_npi: 1 = 0x00000001
2017-09-26 12:04:08 [15922] [6] DEBUG: destination_addr: "43676....."
2017-09-26 12:04:08 [15922] [6] DEBUG: esm_class: 3 = 0x00000003
2017-09-26 12:04:08 [15922] [6] DEBUG: protocol_id: 0 = 0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG: priority_flag: 0 = 0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG: schedule_delivery_time: NULL
2017-09-26 12:04:08 [15922] [6] DEBUG: validity_period: NULL
2017-09-26 12:04:08 [15922] [6] DEBUG: registered_delivery: 0 = 0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG: replace_if_present_flag: 0 = 0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG: data_coding: 0 = 0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG: sm_default_msg_id: 0 = 0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG: sm_length: 23 = 0x00000017
2017-09-26 12:04:08 [15922] [6] DEBUG: short_message:
2017-09-26 12:04:08 [15922] [6] DEBUG:   Octet string at 0x7f227400c460:
2017-09-26 12:04:08 [15922] [6] DEBUG:     len: 23
2017-09-26 12:04:08 [15922] [6] DEBUG:     size: 24
2017-09-26 12:04:08 [15922] [6] DEBUG:     immutable: 0
2017-09-26 12:04:08 [15922] [6] DEBUG:     data: 44 <14 bytes> 73
2017-09-26 12:04:08 [15922] [6] DEBUG:     data: 74 <5 bytes> 39
2017-09-26 12:04:08 [15922] [6] DEBUG:   Octet string dump ends.
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]: throughput (1.00,5.00)
...

```

– Example link enquiry:

```

...
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP[default_smsc]: throughput (0.00,5.00)
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP[default_smsc]: Got PDU:
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP PDU 0x7f2274020790 dump:
2017-09-26 12:13:38 [15922] [6] DEBUG:   type_name: enquire_link
2017-09-26 12:13:38 [15922] [6] DEBUG:   command_id: 21 = 0x00000015
2017-09-26 12:13:38 [15922] [6] DEBUG:   command_status: 0 = 0x00000000
2017-09-26 12:13:38 [15922] [6] DEBUG:   sequence_number: 90764 = 0x0001628c
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP[default_smsc]: Sending PDU:
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP PDU 0x7f2274025070 dump:
2017-09-26 12:13:38 [15922] [6] DEBUG:   type_name: enquire_link_resp
2017-09-26 12:13:38 [15922] [6] DEBUG:   command_id: 2147483669 = 0x80000015
2017-09-26 12:13:38 [15922] [6] DEBUG:   command_status: 0 = 0x00000000
2017-09-26 12:13:38 [15922] [6] DEBUG:   sequence_number: 90764 = 0x0001628c

```

```

2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP[default_smsc]: throughput (0.00,5.00)
...

```

### 7.29.2.2 SMS Box (PRX node of NGCP)

On the PRX node you can see a **process** named "**smsbox**". This process has a **listening port** assigned to it: 13002, that is the communication port towards the *Bearer Box* component running on LB nodes.

The *ngcp-service* tool also shows the *smsbox* process in its summary information:

```

$ ngcp-service summary
Ok Service                               Managed   Started   Status
--  -----
...
  kannel-smsbox                           managed   by-ha     active
...

```

The following log files can provide information about the operation of *SMS Box*:

- sent and received messages using the API of WEB node: `/var/log/kannel/smsbox.log`

---

#### Note

Sent and received message examples shown here do not contain the full phone number and content for confidentiality reason.

---

– Example sent message:

```

...
2017-09-26 12:16:42 [22763] [2] DEBUG: HTTP: Creating HTTPClient for '192.168.1.2'.
2017-09-26 12:16:42 [22763] [2] DEBUG: HTTP: Created HTTPClient area 0x7f5dcc000ad0.
2017-09-26 12:16:42 [22763] [3] INFO: smsbox: Got HTTP request </cgi-bin/sendsms> from ↔
  <192.168.1.3>
2017-09-26 12:16:42 [22763] [3] INFO: sendsms used by <sipwise>
2017-09-26 12:16:42 [22763] [3] INFO: sendsms sender:<sipwise:43668.....> ↔
  (192.168.1.3) to:<43676.....> msg:<...>
2017-09-26 12:16:42 [22763] [3] DEBUG: Stored UUID ab95eb45-1ec0-4932-9863-1a95609a025f
2017-09-26 12:16:42 [22763] [3] DEBUG: message length 52, sending 1 messages
2017-09-26 12:16:42 [22763] [3] DEBUG: Status: 202 Answer: <Sent.>
2017-09-26 12:16:42 [22763] [3] DEBUG: Delayed reply - wait for bearerbox
2017-09-26 12:16:42 [22763] [0] DEBUG: Got ACK (0) of ab95eb45-1ec0-4932-9863-1 ↔
  a95609a025f
2017-09-26 12:16:42 [22763] [0] DEBUG: HTTP: Destroying HTTPClient area 0x7f5dcc000ad0.
2017-09-26 12:16:42 [22763] [0] DEBUG: HTTP: Destroying HTTPClient for '192.168.1.3'.

```

...

## – Example received message:

...

```

2017-09-26 11:59:45 [22763] [5] INFO: Starting to service <...message content...> from ←
  <+43676-----> to <+43668----->
2017-09-26 11:59:45 [22763] [10] DEBUG: Queue contains 0 pending requests.
2017-09-26 11:59:45 [22763] [10] DEBUG: HTTPS URL; Using SSL for the connection
2017-09-26 11:59:45 [22763] [10] DEBUG: Parsing URL `https://192.168.1.2:1443/ ←
  internalsms/receive?auth_token=fNLosMgwdNURKvEfFMm9
&timestamp=2017-09-26+09:59:45&from=%2B43676-----&to=%2B43668-----&charset=UTF-8& ←
  coding=0&text=...':
2017-09-26 11:59:45 [22763] [10] DEBUG:   Scheme: https://
2017-09-26 11:59:45 [22763] [10] DEBUG:   Host: 192.168.1.2
2017-09-26 11:59:45 [22763] [10] DEBUG:   Port: 1443
2017-09-26 11:59:45 [22763] [10] DEBUG:   Username: (null)
2017-09-26 11:59:45 [22763] [10] DEBUG:   Password: (null)
2017-09-26 11:59:45 [22763] [10] DEBUG:   Path: /internalsms/receive
2017-09-26 11:59:45 [22763] [10] DEBUG:   Query: auth_token=fNLosMgwdNURKvEfFMm9& ←
  timestamp=2017-09-26+09:59:45&from=%2B43676-----
&to=%2B43668-----&charset=UTF-8&coding=0&text=...
2017-09-26 11:59:45 [22763] [10] DEBUG:   Fragment: (null)
2017-09-26 11:59:45 [22763] [10] DEBUG: Connecting nonblocking to <192.168.1.2>
2017-09-26 11:59:45 [22763] [10] DEBUG: HTTP: Opening connection to '192.168.1.2:1443' ( ←
  fd=31).
2017-09-26 11:59:45 [22763] [10] DEBUG: Socket connecting
2017-09-26 11:59:45 [22763] [9] DEBUG: Get info about connecting socket
2017-09-26 11:59:45 [22763] [9] DEBUG: HTTP: Sending request:
2017-09-26 11:59:45 [22763] [9] DEBUG: Octet string at 0x7f5dbc00f470:
2017-09-26 11:59:45 [22763] [9] DEBUG:   len: 382
2017-09-26 11:59:45 [22763] [9] DEBUG:   size: 1024
2017-09-26 11:59:45 [22763] [9] DEBUG:   immutable: 0
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 47 45 54 20 2f 69 6e 74 65 72 6e 61 6c 73 ←
  6d 73   GET /internalsms
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 2f 72 65 63 65 69 76 65 3f 61 75 74 68 5f ←
  74 6f   /receive?auth_to
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 6b 65 6e 3d ... ←
  ken=
  ... 20 48 54 54 50 2f 31 2e 31 ←
  0d 0a   HTTP/1.1..
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 ←
  65 70   Connection: keep
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 2d 61 6c 69 76 65 0d 0a 55 73 65 72 2d 41 ←
  67 65   -alive..User-Age
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 6e 74 3a 20 4b 61 6e 6e 65 6c 2f 31 2e 34 ←
  2e 34   nt: Kannel/1.4.4

```

```

2017-09-26 11:59:45 [22763] [9] DEBUG: data: 0d 0a 48 6f 73 74 3a 20 31 39 32 2e 31 36 ←
      38 2e ..Host: 192.168.
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 31 2e 32 3a 31 34 34 33 0d 0a 0d 0a ←
      1.2:1443....
2017-09-26 11:59:45 [22763] [9] DEBUG: Octet string dump ends.
2017-09-26 11:59:45 [22763] [9] DEBUG: HTTP: Status line: <HTTP/1.1 200 OK>
2017-09-26 11:59:45 [22763] [9] DEBUG: HTTP: Received response:
2017-09-26 11:59:45 [22763] [9] DEBUG: Octet string at 0x7f5dbc006970:
2017-09-26 11:59:45 [22763] [9] DEBUG: len: 333
2017-09-26 11:59:45 [22763] [9] DEBUG: size: 1024
2017-09-26 11:59:45 [22763] [9] DEBUG: immutable: 0
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 53 65 72 76 65 72 3a 20 6e 67 69 6e 78 0d ←
      0a 44 Server: nginx..D
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 61 74 65 3a 20 54 75 65 2c 20 32 36 20 53 ←
      65 70 ate: Tue, 26 Sep
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 20 32 30 31 37 20 30 39 3a 35 39 3a 34 35 ←
      20 47 2017 09:59:45 G
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 4d 54 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 ←
      70 65 MT..Content-Type
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 3a 20 74 65 78 74 2f 68 74 6d 6c 3b 20 63 ←
      68 61 : text/html; cha
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 72 73 65 74 3d 75 74 66 2d 38 0d 0a 43 6f ←
      6e 74 rset=utf-8..Cont
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 30 0d ←
      0a 43 ent-Length: 0..C
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 ←
      70 2d onnection: keep-
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 61 6c 69 76 65 0d 0a 53 65 74 2d 43 6f 6f ←
      6b 69 alive..Set-Cooki
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 65 3a 20 6e 67 63 70 5f 70 61 6e 65 6c 5f ←
      73 65 e: ngcp_panel_se
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 73 73 69 6f 6e 3d 34 35 30 32 64 64 66 65 ←
      31 62 ssion=4502ddfelb
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 63 31 65 33 39 30 65 30 64 36 66 39 64 34 ←
      37 30 c1e390e0d6f9d470
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 35 30 37 62 64 64 33 61 65 32 36 62 64 63 ←
      3b 20 507bdd3ae26bdc;
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 70 61 74 68 3d 2f 3b 20 65 78 70 69 72 65 ←
      73 3d path=/; expires=
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 54 75 65 2c 20 32 36 2d 53 65 70 2d 32 30 ←
      31 37 Tue, 26-Sep-2017
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 20 31 30 3a 35 39 3a 34 35 20 47 4d 54 3b ←
      20 48 10:59:45 GMT; H
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 74 74 70 4f 6e 6c 79 0d 0a 58 2d 43 61 74 ←
      61 6c ttpOnly..X-Catal
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 79 73 74 3a 20 35 2e 39 30 30 37 35 0d 0a ←
      53 74 yst: 5.90075..St
2017-09-26 11:59:45 [22763] [9] DEBUG: data: 72 69 63 74 2d 54 72 61 6e 73 70 6f 72 74 ←

```

```

    2d 53    rict-Transport-S
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 65 63 75 72 69 74 79 3a 20 6d 61 78 2d 61 ↔
    67 65    ecurity: max-age
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 3d 31 35 37 36 38 30 30 30 0d 0a 0d 0a ↔
    =15768000....
2017-09-26 11:59:45 [22763] [9] DEBUG: Octet string dump ends.
2017-09-26 11:59:45 [22763] [6] WARNING: Tried to set Coding field, denied.
2017-09-26 11:59:45 [22763] [6] INFO: No reply sent, denied.
2017-09-26 11:59:55 [22763] [9] DEBUG: HTTP: Server closed connection, destroying it ↔
    <192.168.1.2:1443:1::><0x7f5db000b20><fd:31>.
...

```

- short log of sent/received messages: `/var/log/kannel/smsbox-access.log`

```

...
2017-09-26 12:39:18 SMS HTTP-request sender:+43680----- request: '' url: 'https ↔
    ://192.168.1.2:1443/internalsms/receive?
auth_token=fNLosMgwdNUrKvEfFMm9&timestamp=2017-09-26+10:39:18&from=%2B43680-----&to=%2 ↔
    B43668-----&charset=UTF-8&coding=0
&text=<...message content...>' reply: 200 '<< successful >>'
...
2017-09-26 12:41:54 send-SMS request added - sender:sipwise:43668----- 192.168.1.3 ↔
    target:43680----- request: '<...message content...>'
...

```

### 7.29.3 REST API

Handling of short messages from the user perspective happens with the help of NGCP's REST API. There is a dedicated resource: `https://<IP of WEB node>:1443/api/sms` that allows you to:

- **Get a list of sent and received messages.** This is achieved by sending a GET request on the `/api/sms` collection, as in the following example:

```

curl -i -X GET -H 'Connection: close' --cert NGCP-API-client-certificate.pem \
    --cacert ca-cert.pem 'https://example.org:1443/api/sms/?page=1&rows=10'

```

- **Retrieve an SM** (both sent and received). This is achieved by sending a GET request for a specific `/api/sms/id` item, as in the following example:

```

curl -i -X GET -H 'Connection: close' --cert NGCP-API-client-certificate.pem \
    --cacert ca-cert.pem 'https://example.org:1443/api/sms/1'

```

- **Send a new message** from a local subscriber. This is achieved by sending a POST request for the `/api/sms` collection, as in the following example:

```
curl -i -X POST -H 'Connection: close' -H 'Content-Type: application/json' \  
  --cert NGCP-API-client-certificate.pem --cacert ca-cert.pem \  
  'https://example.org:1443/api/sms/' --data-binary '{"callee" : "43555666777", \  
  "subscriber_id" : 4, "text" : "test"}'
```

As always, the full documentation of the REST API resources is available on the admin web interface of NGCP: `https://<IP of WEB node>:1443/api/#sms`

## 7.30 Time sets management

### 7.30.1 Time sets specifications and data description

The Sipwise C5 provides administrative WEB and API interface to manage time sets.

Supported fields, input and output format are based on **iCalendar EVENT** specification.

Not all iCalendar and EVENT properties are supported, but those that are used for time points and periods definition or stated mandatory by specification:

- CALENDAR supported properties:

- NAME

- EVENT supported properties:

- SUMMARY

- DTSTART

- DTEND

- RRULE

Important to mention that current implementation does not support these EVENT properties:

- DTSTAMP ( UID is used in generated calendar ics file, both UID and DTSTAMP are ignored during uploading calendar file );
- DURATION ( DTEND is used );
- RDATE
- EXDATE
- PRIORITY

Main EVENT property, that is used for time points and periods definition is RRULE. Current time sets implementation supports all properties described in the [RRULE specification](#) except WKST.

Default value for week start is MO (Monday).

### 7.30.2 Web interface for the time sets

Time sets management section is provided in two variants. One is main time sets management section and other is a chapter on reseller details page. Variants have minor differences. Functionality will be explained using time sets dedicated interface. Differences will be explained below.

Time set can be created using creation form. On time sets management interface press button "Create Time Set Entry":

The screenshot shows the Sipwise NGCP Dashboard interface. At the top, there is a navigation bar with the logo and the text "sip:wise NGCP Dashboard". Below the logo, there are several menu items: Home, Documentation, Monitoring & Statistics, Tools, and Settings. The main content area is titled "Time Set". Below the title, there are two buttons: "Back" and "Create Time Set Entry". The "Create Time Set Entry" button is highlighted with a red box. Below the buttons, there is a search bar and a table with the following data:

| # | Reseller               | Name  |
|---|------------------------|-------|
| 1 | api_test test reseller | test2 |

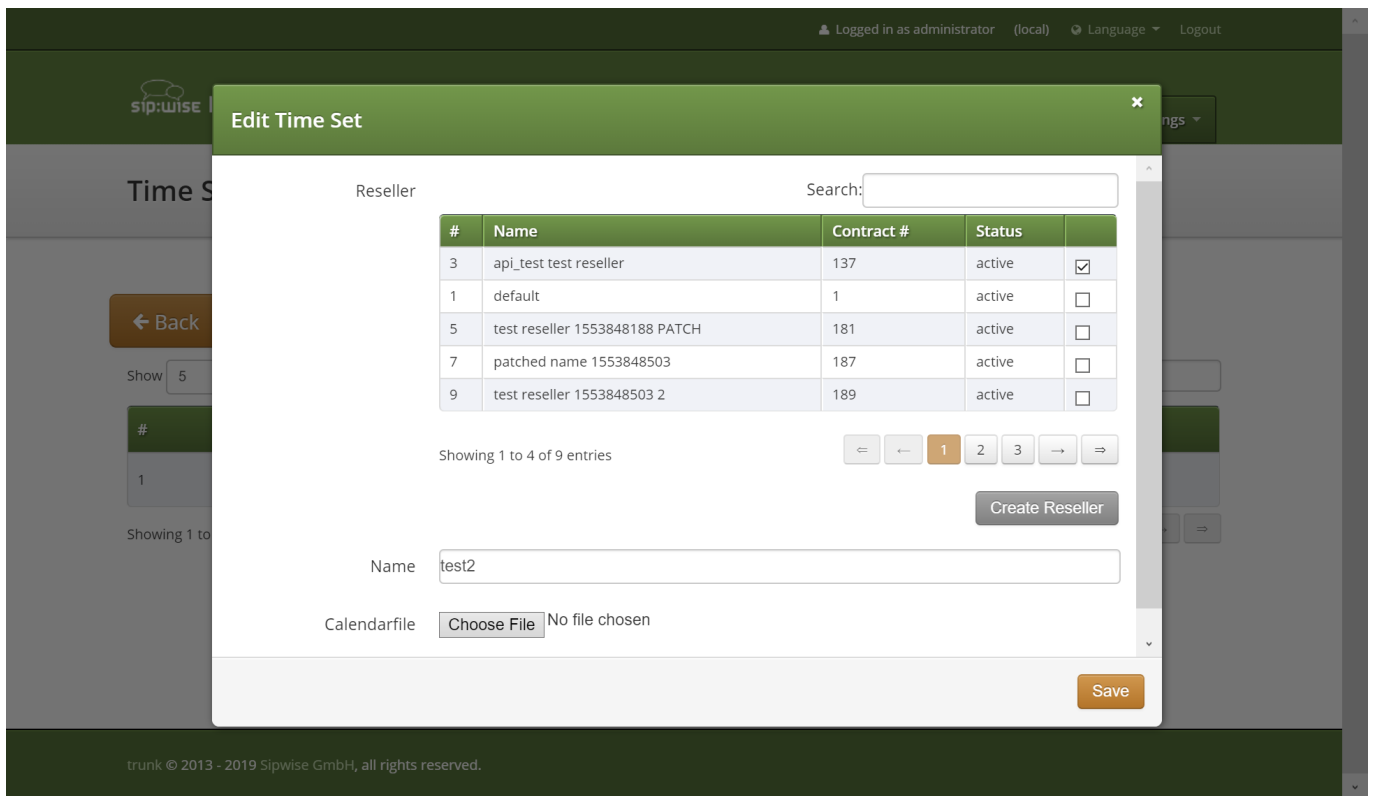
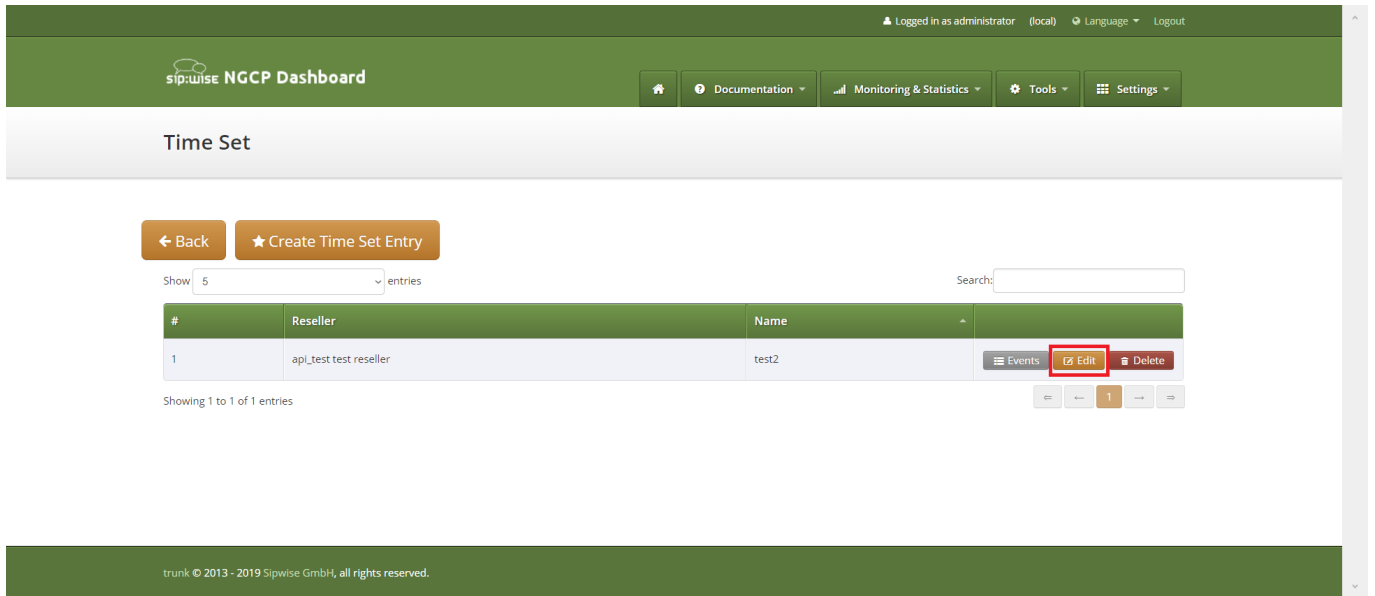
Below the table, it says "Showing 1 to 1 of 1 entries". At the bottom of the dashboard, there is a footer with the text "trunk © 2013 - 2019 Sipwise GmbH, all rights reserved."

"Reseller" field is mandatory.

"Name" field should be defined, if iCalendar (ics) file is not going to be uploaded or file doesn't have NAME property for the CALENDAR entry.

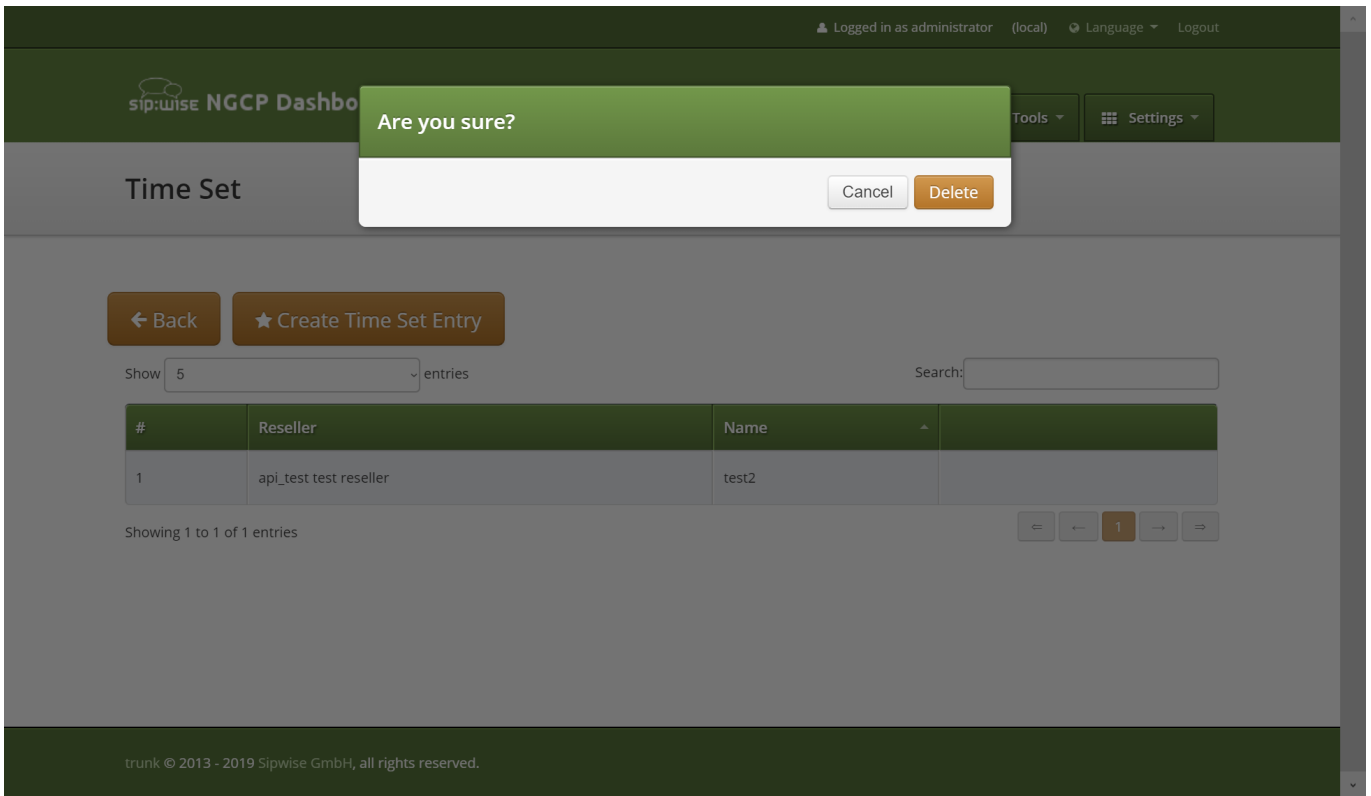
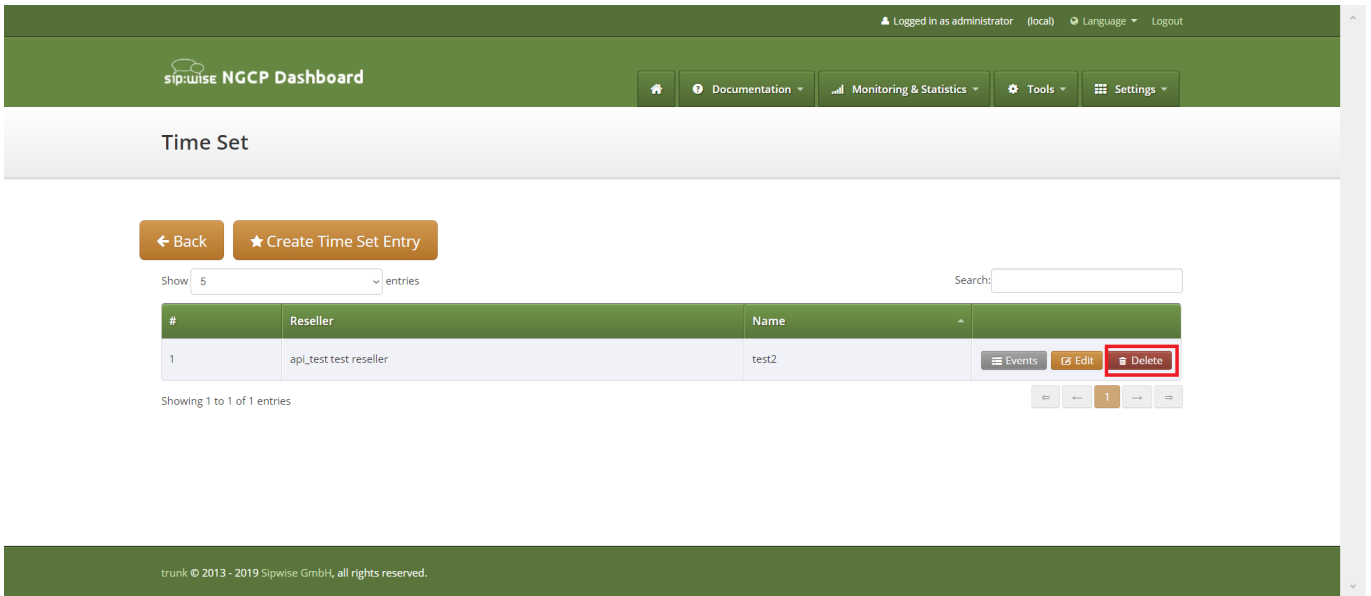
If both NAME in the uploaded iCalendar (ics) file and form field "Name" aren't empty then value from the form field will be taken.

Created time set can be modified:



or deleted:





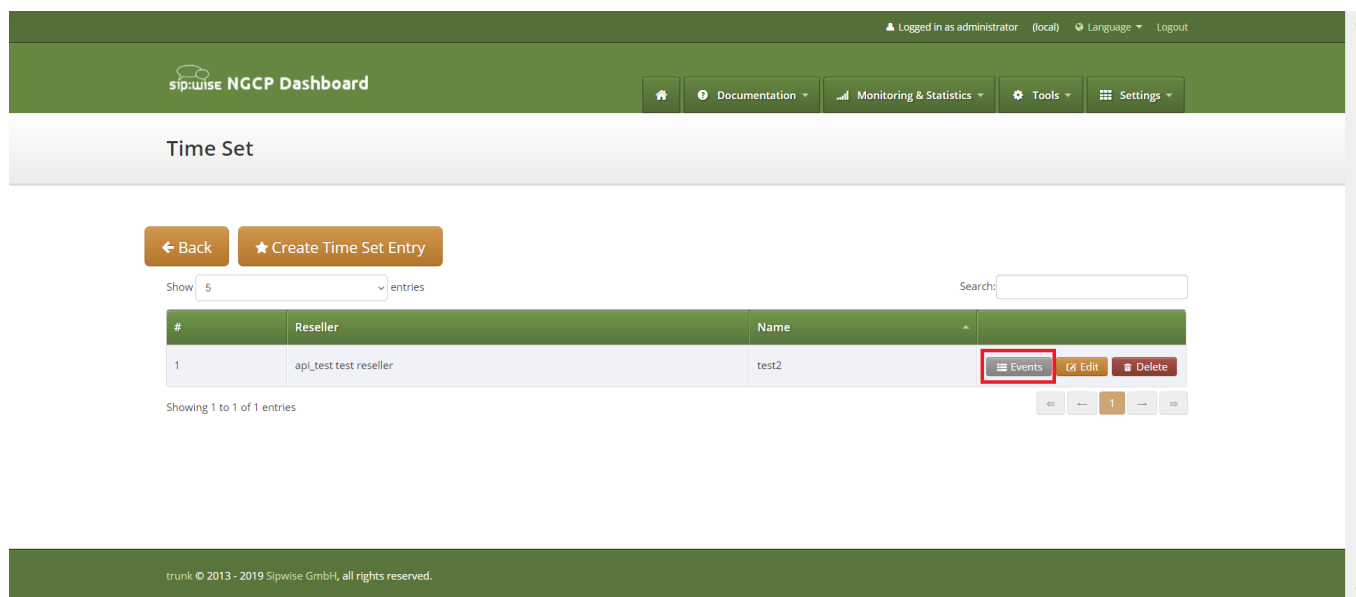
**Note**

If calendar ics file will be uploaded to edit time set, all presented events will be deleted and events from the uploaded file will be added after it.

**7.30.3 Web interface for the time set events**

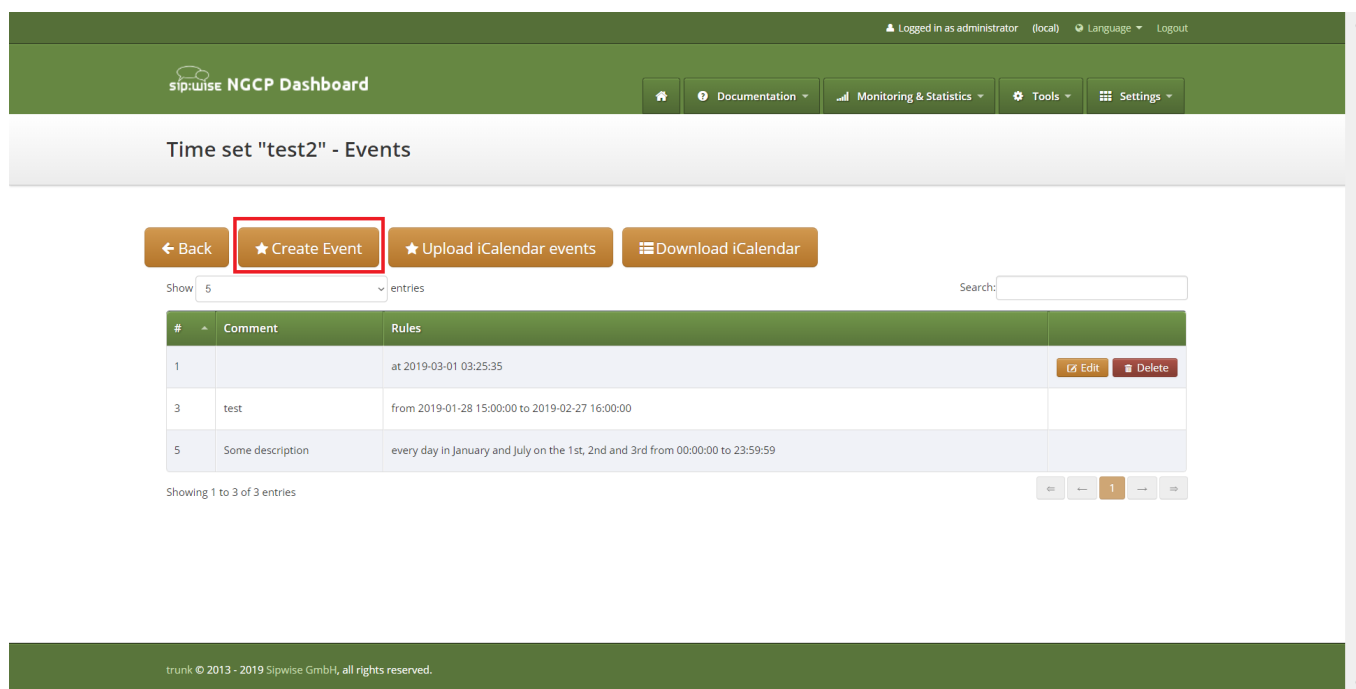
Time set can contain set of events. Each time set event will be used to generate CALENDAR EVENT entry in the generated iCalendar file. So all fields in the time set event forms represent properties of the iCalendar EVENT component.

To manage time set events press "Events" button against proper time set.

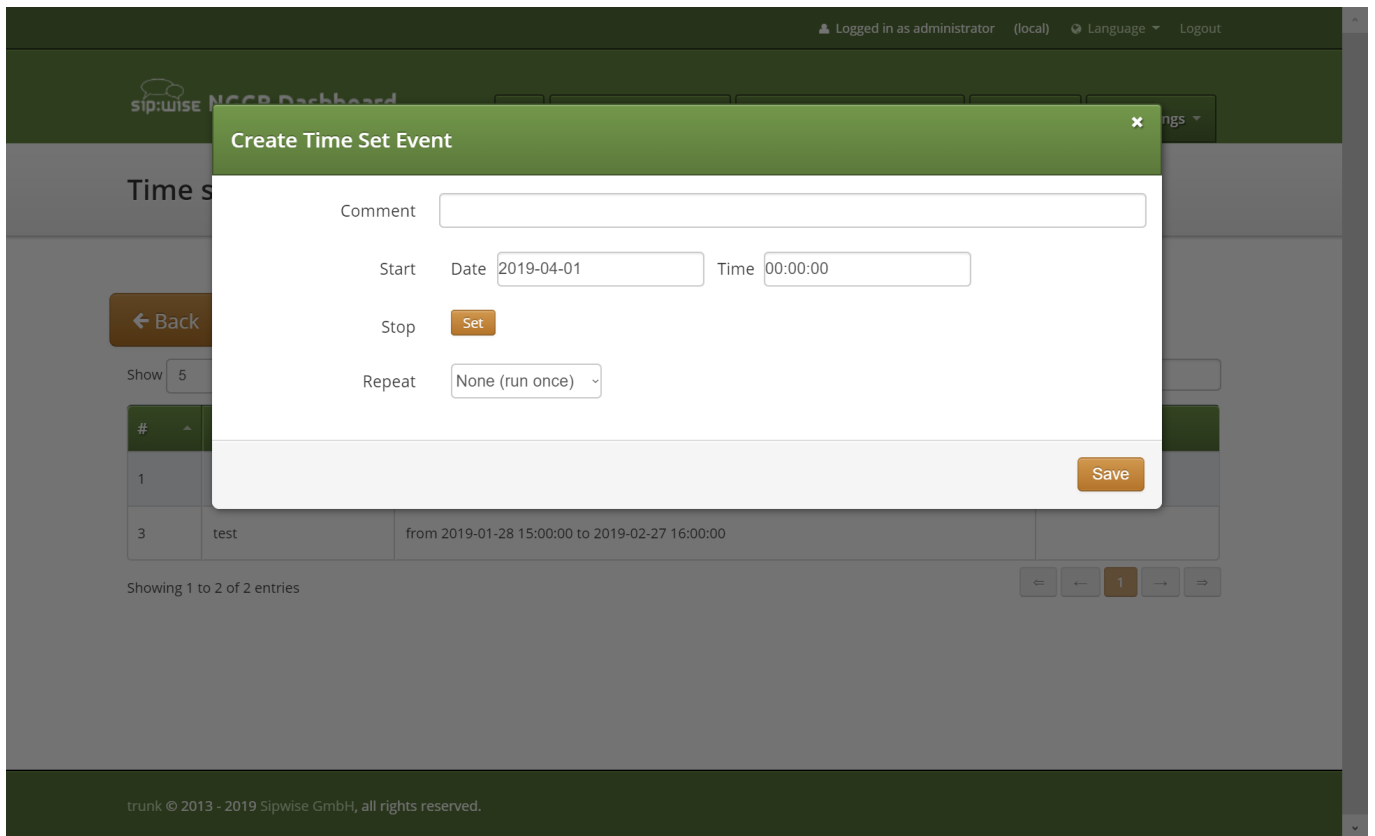


Events management section will appear:

To create event press "Create Event" button.



Form to create event will be shown:



The screenshot shows the 'Create Time Set Event' modal form in the Sipwise NCCP Dashboard. The form is overlaid on a background of a 'Time Set' table. The modal contains the following fields and controls:

- Comment:** A text input field.
- Start:** A sub-section with two input fields: 'Date' (containing '2019-04-01') and 'Time' (containing '00:00:00').
- Stop:** A button labeled 'Set'.
- Repeat:** A dropdown menu currently showing 'None (run once)'.
- Save:** A button at the bottom right of the modal.

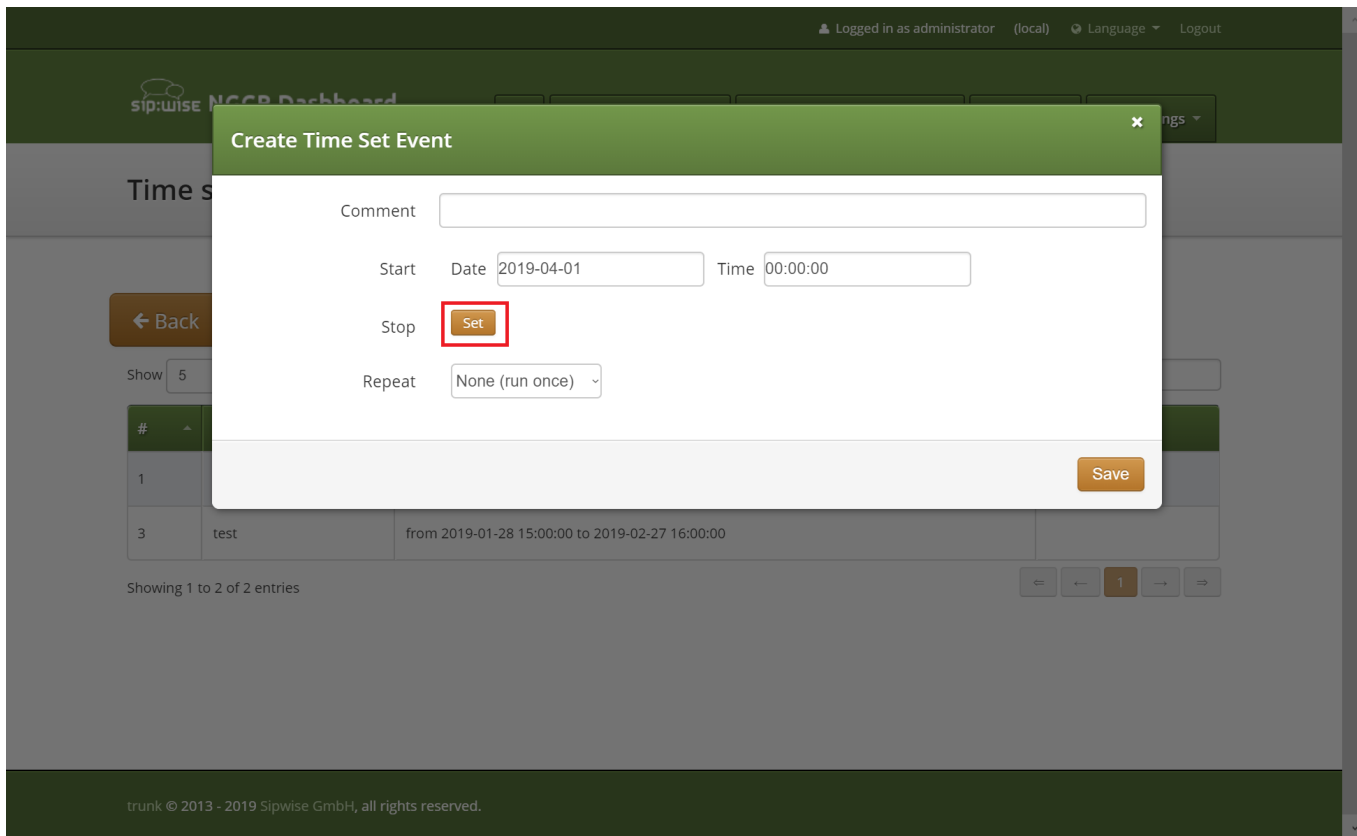
The background table shows a list of time set events. The first row is partially visible with the number '1'. The second row shows an event with ID '3', comment 'test', and a duration 'from 2019-01-28 15:00:00 to 2019-02-27 16:00:00'. The table is showing 1 to 2 of 2 entries.

### 7.30.3.1 Time set event form fields explanation

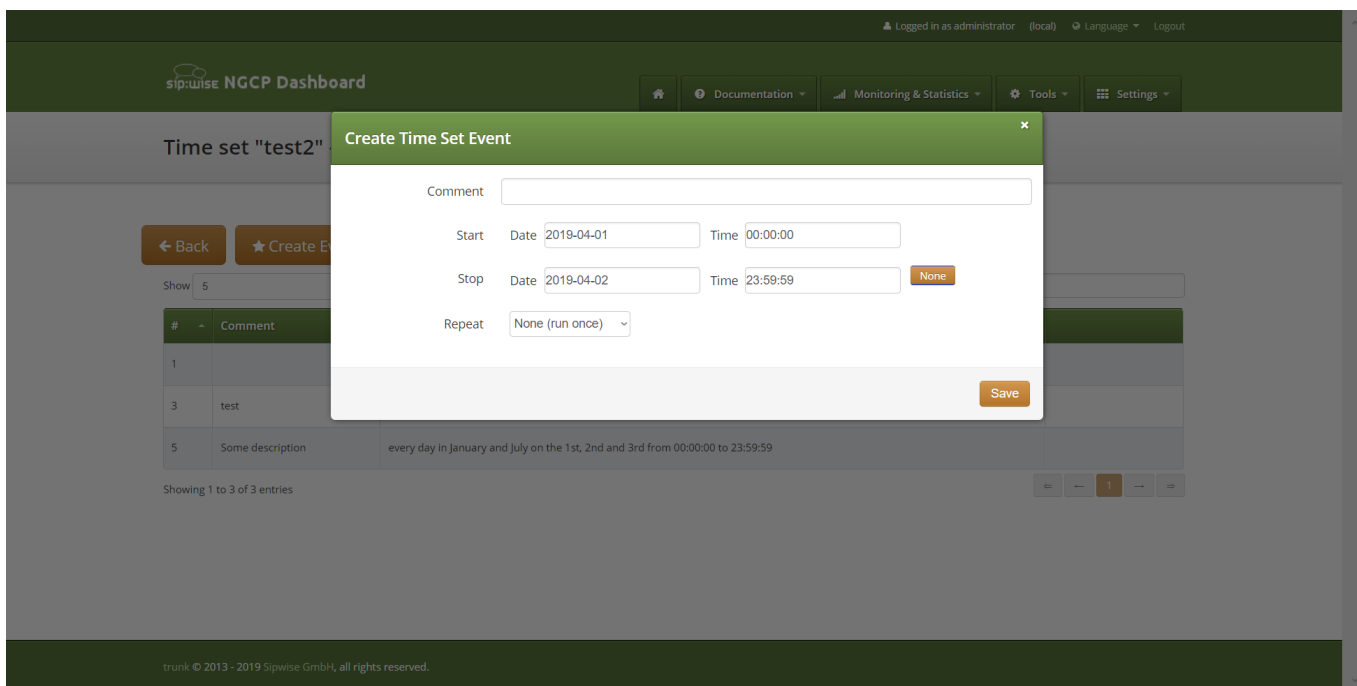
"Start" field reflects DTSTART property of the EVENT. "Start" is mandatory and by default is set to the start of the current day. "Start" value format is datetime.

"Stop" field reflects DTEND property of the EVENT. For the events within recurrence "Stop" will define duration of each iteration.

To specify "Stop" datetime, press button "Set".



Fields to enter DTEND date and time will appear:



To return "Stop" field to the empty value press button "None". Value in the form fields will be preserved, but newly created EVENT will have empty DTEND property.

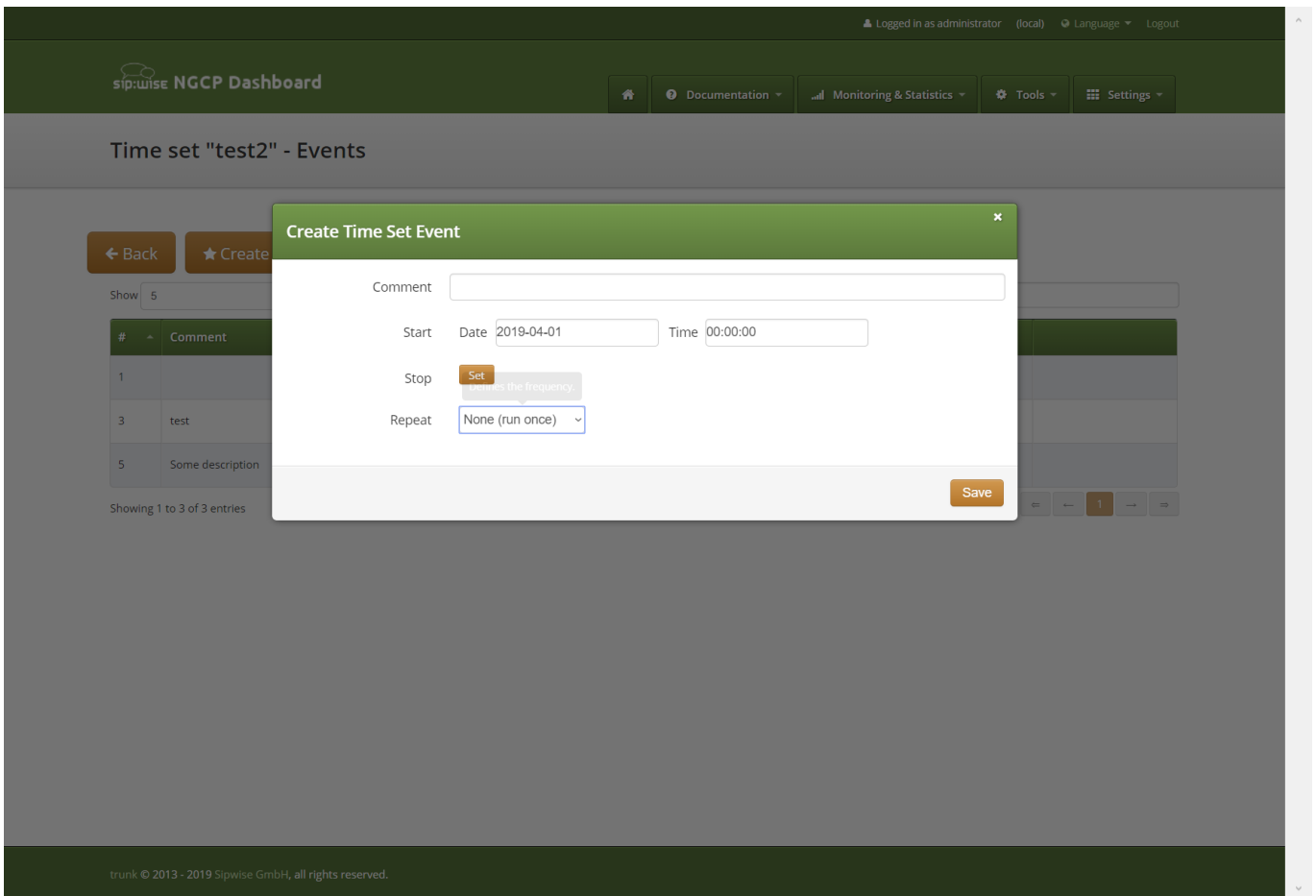
Other fields in the form are optional. Most of them aren't visible by default and will be shown if requested by user or required by

data into other fields.

RRULE property of the EVENT is a recurrence rule and defines set of the iterations for the EVENT.

To customize recurrence rule for the EVENT select proper repetition unit for the "Repeat" form field. Input field for the recurrence interval will appear left to the frequency select.

According to the selected unit, FREQ property of the EVENT RRULE will be set to one of the: SECONDLY, MINUTELY, HOURLY, DAILY, WEEKLY, MONTHLY, YEARLY.

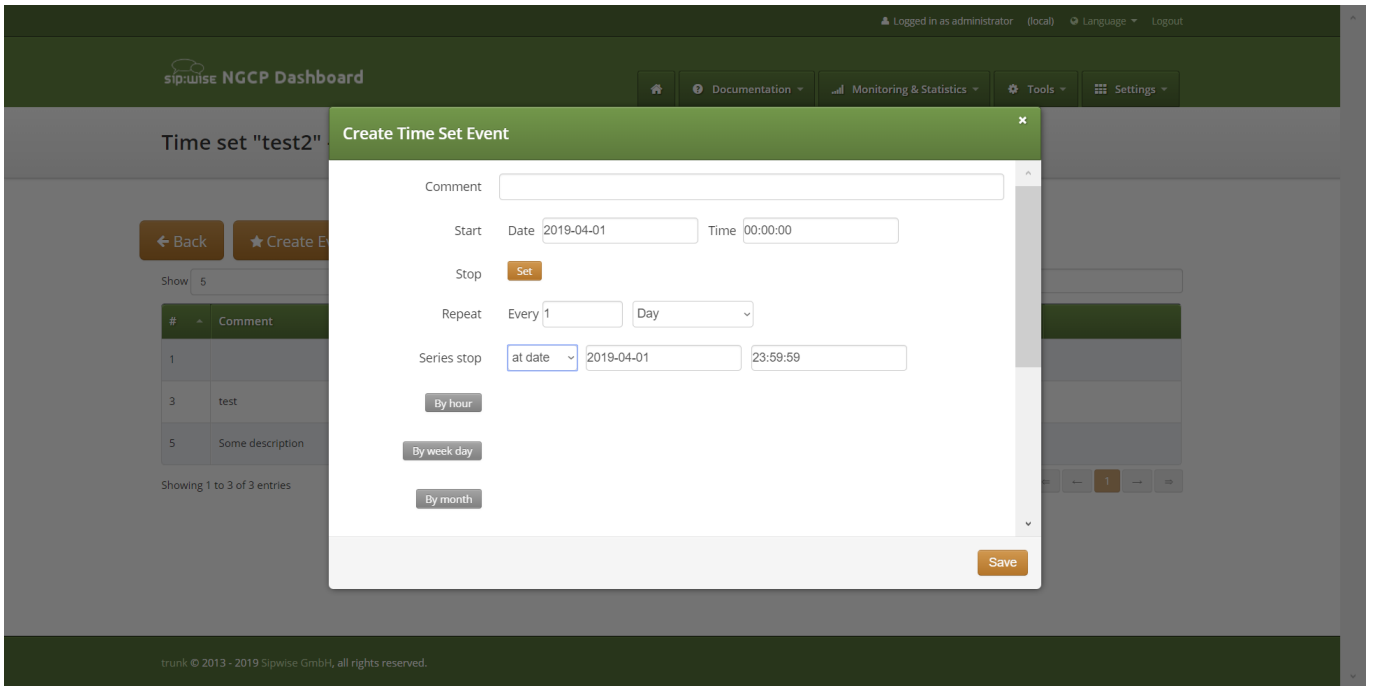


The screenshot shows the Sipwise NGCP Dashboard interface. At the top, there is a navigation bar with 'sip:wise NGCP Dashboard' and several menu items: 'Documentation', 'Monitoring & Statistics', 'Tools', and 'Settings'. The main content area is titled 'Time set "test2" - Events'. A modal dialog box titled 'Create Time Set Event' is open in the center. The dialog box contains the following fields and options:

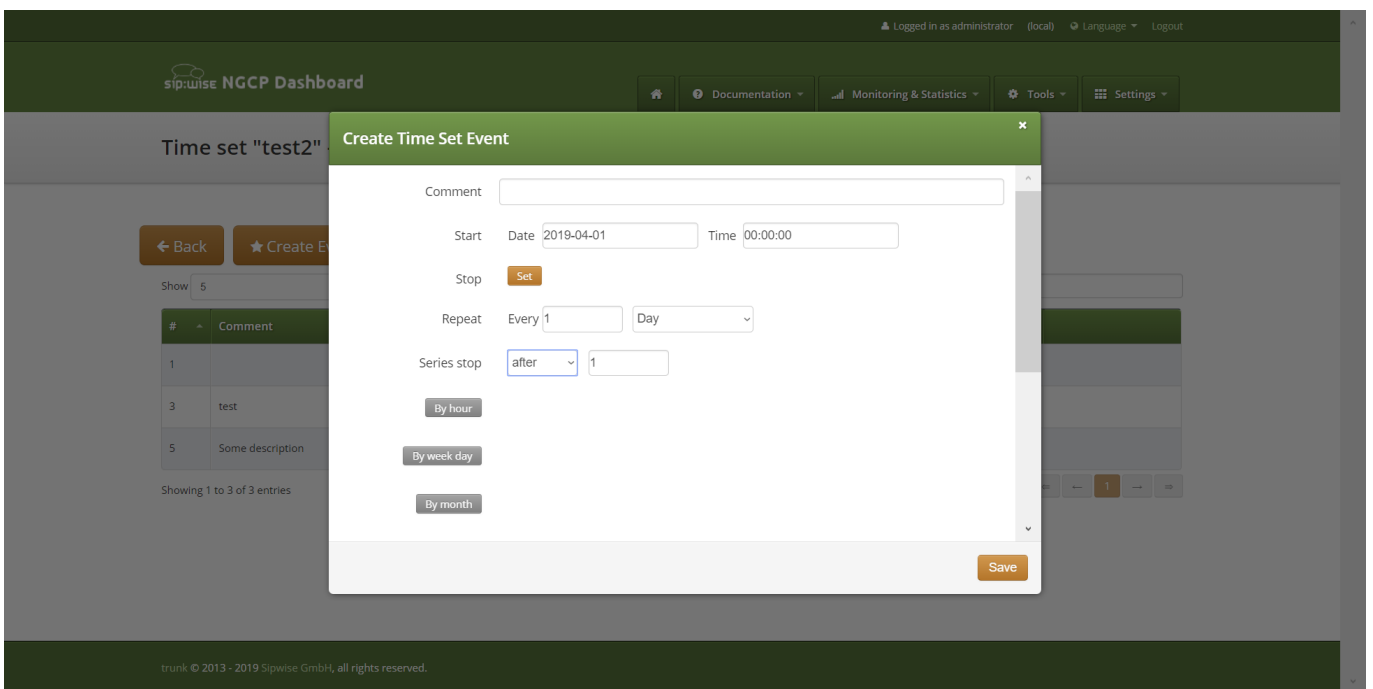
- Comment:** A text input field.
- Start:** A 'Date' field with the value '2019-04-01' and a 'Time' field with the value '00:00:00'.
- Stop:** A dropdown menu with the option 'Set as the frequency' selected.
- Repeat:** A dropdown menu with the option 'None (run once)' selected.
- Save:** An orange button at the bottom right of the dialog.

In the background, a table of events is visible with columns for '#', 'Comment', and 'Showing 1 to 3 of 3 entries'. The footer of the dashboard reads 'trunk © 2013 - 2019 Sipwise GmbH, all rights reserved.'

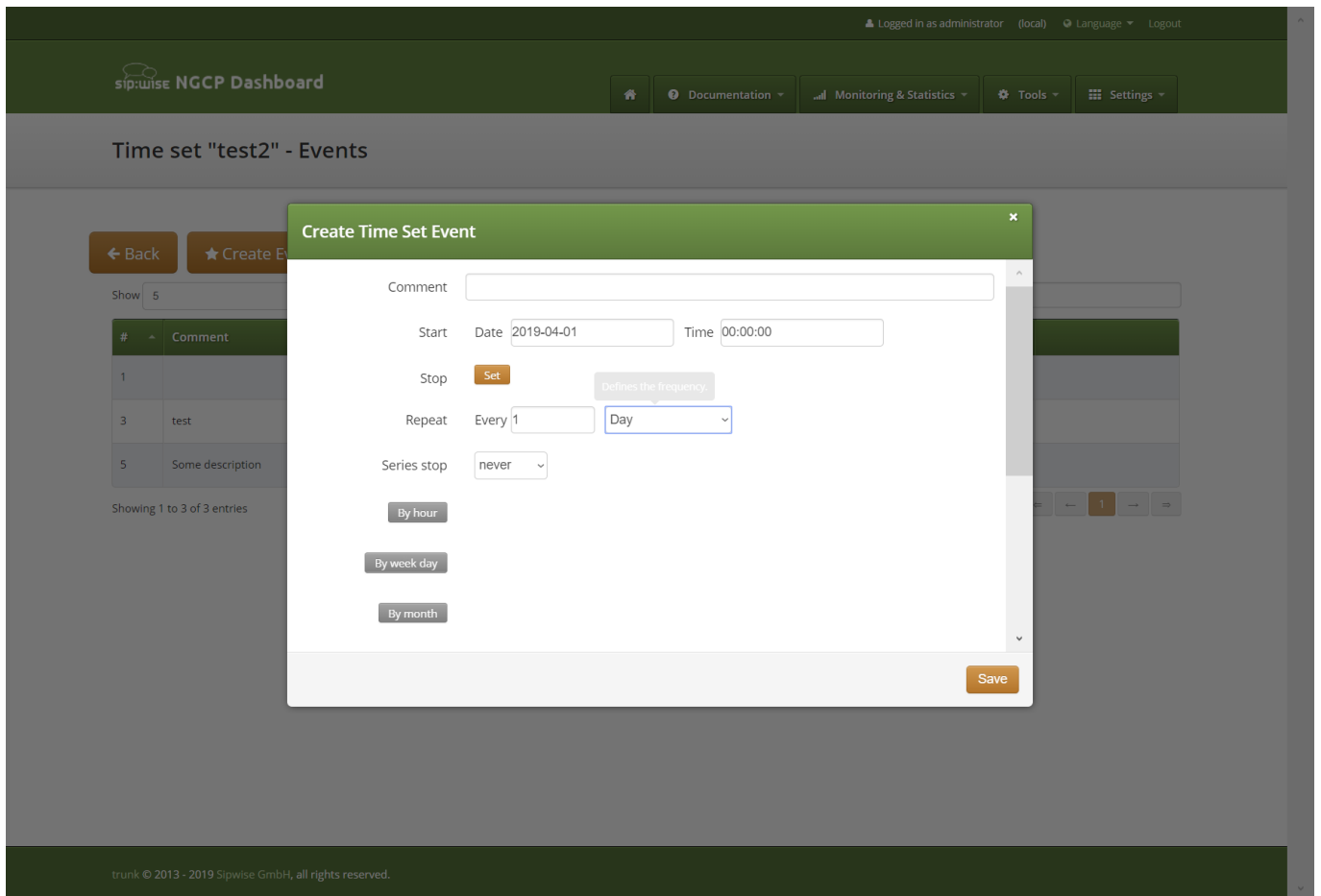
To specify end of the EVENT iterations, select "Series stop" value. For the "at date" option will be shown input for the date and time that will define UNTIL property of the EVENT RRULE.



"after" option, respectively, will put entered value to the COUNT property of the EVENT RRULE.



Form fields "By hour", "By week day", "By month", "By month day", "By set position", "By week number", "By year day", "By second" and "By minute" aren't shown by default. To enter value to any of these fields press according button on the left. Button with field name is grayed off when corresponding EVENT property is empty.



The screenshot displays the Sipwise NGCP Dashboard interface. At the top, the user is logged in as an administrator. The main navigation bar includes links for Documentation, Monitoring & Statistics, Tools, and Settings. The current page is titled "Time set 'test2' - Events". A modal dialog box titled "Create Time Set Event" is open, allowing the user to configure a new event. The dialog includes a text input for a comment, a date and time selector for the start (set to 2019-04-01 00:00:00), a "Stop" button (currently orange), a "Repeat" section with a frequency of "Every 1 Day", and a "Series stop" dropdown set to "never". Below these are three frequency selection buttons: "By hour", "By week day", and "By month". A "Save" button is located at the bottom right of the dialog. In the background, a table lists existing events with columns for ID and Comment.

| # | Comment          |
|---|------------------|
| 1 |                  |
| 3 | test             |
| 5 | Some description |

When gray button with field name is pressed, field input control appears on the right. In the same time button with field name becomes orange, indicating that field value will be saved for the EVENT.

Fields with checkboxes controls have auxiliary button "Invert selection". When button "Invert selection" is pressed currently empty checkboxes become selected and currently selected checkboxes become empty.

trunk © 2013 - 2019 Sipwise GmbH, all rights reserved.

When form data will be saved, checkboxes values will be saved as coma separated numbers.

BYxxx RRULE properties expand or limit behavior of the FREQ according to the table in the [RRULE specification](#).

Field "By week day" has two variants of the input: checkbox for each week day and text input. Text input can be used, if "By week day" value is more complex than just list of week days, separated by coma, for example for FREQ MONTHLY value "2TH,-3FR" in the "By week day" will mean second Thursday from the month start and third Friday from the month end in every month. Such value can't be presented as checkboxes selection.

Fields "By set position" and "By year day" are text inputs. Value format for these fields is set of the [+/-]NUMBER values, separated by comma.

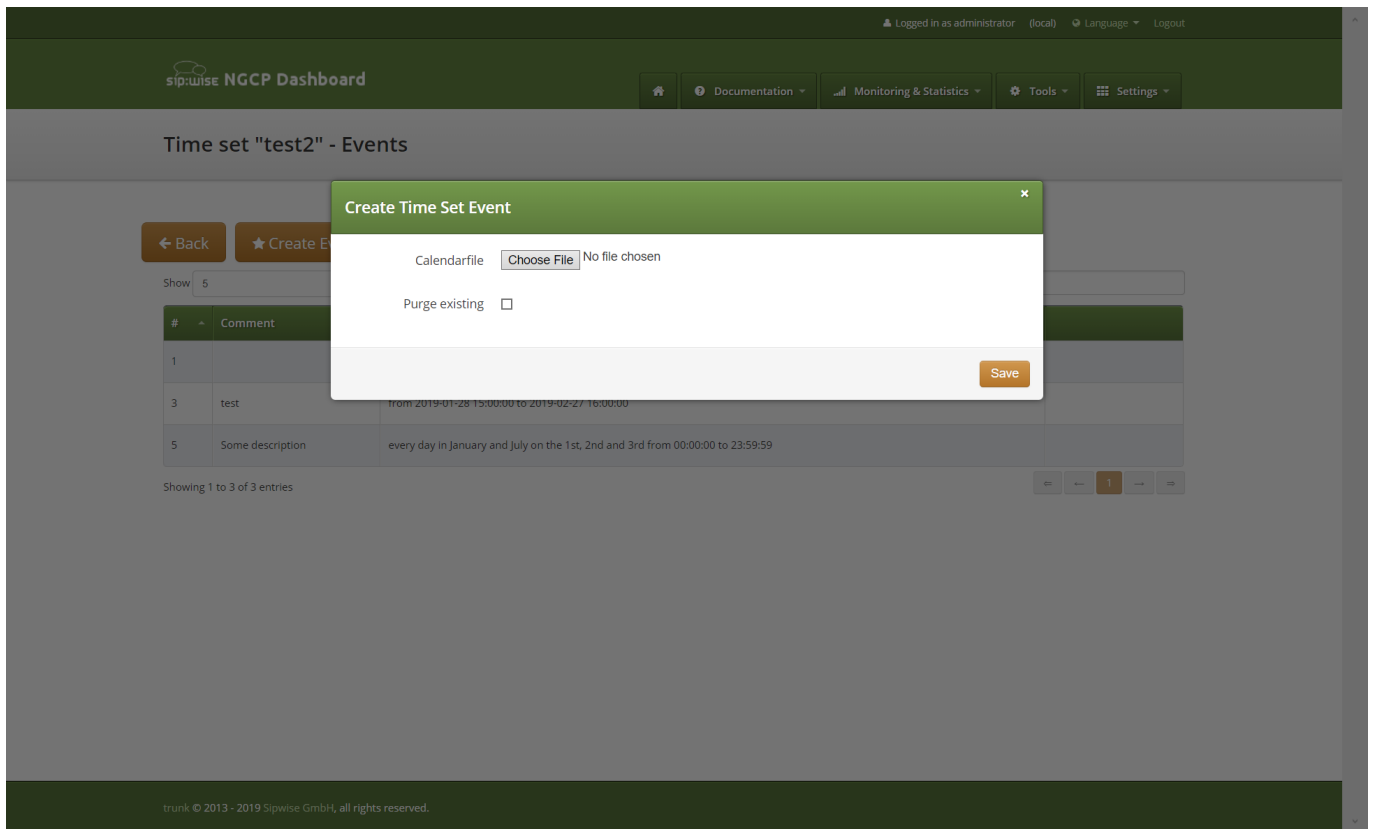
For the "By year day" minus sign in front of year day number means that this day should be taken by number from the end of the year.

For the "By set position" minus sign in front of the position of the iteration means that the iteration should be taken by number from the end of the generated iterations sequence.

After new event created, event will appear in time set event list. It will have column with rrule text description, buttons to request event edit form or event deletion.

In events list section all events can be redefined uploading ics iCalendar file:





If "Purge existing" option is selected, all existing time set events will be removed before creation of the events from the uploaded file.

To download iCalendar ics file of the time set, press "Download iCalendar".

#### 7.30.4 Web interface for time set related to reseller

Reseller details page provides list of the time sets connected to the reseller and allows create, edit, delete and download time set and has link to the time set events section:

Reseller Details for api\_test test reseller

← Back Preferences Expand Groups

Reseller Base Information

Reseller Contract

Reseller Contact

Administrator Logins

Domains

Billing Profiles

Billing Networks

Profile Packages

Customers

Branding

Invoice Templates

Phonebook

Time sets

★ Create Time Set

Show 5 entries Search:

| # | Name  | Events | Download | Edit | Delete |
|---|-------|--------|----------|------|--------|
| 1 | test2 |        |          |      |        |

Showing 1 to 1 of 1 entries

trunk © 2013 - 2019 Sipwise GmbH, all rights reserved.

In difference to the main time set interface, iCalendar ics file for the time set can be downloaded from the time set list pressing "Download" button.

Creation form doesn't have "Reseller" field and is processed in context of the current reseller.

### 7.30.5 REST API

Time sets management is possible using API REST entry point `/api/timesets/`.

Time sets API has possibility to get and return information both as "application/json" data and as "text/calendar" file.

To create time set with events full specification of the all fields in json format can be used:

```
curl --request POST --user administrator:administrator --header Prefer: return=representation --header Content-Type: application/json https://127.0.0.1:1443/api/timesets/ --data {"reseller": "api_test", "name": "api_test_timeset_name1", "times": [{"start": "1971-01-01 00:00:01", "until": "1997-01-01 00:00:01"}]}
```

```
23:59:59", "end": "2020-12-31 23:59:59"}]]}
```

Also time set and events can be uploaded as ics iCalendar file:

```
curl --request POST --user administrator:administrator --header Prefer: return=representation
--header Content-Type: multipart/form-data https://127.0.0.1:1443/api/timesets/ --form
json={"reseller_id":3, "name": "unique_name"} --form calendarfile=@/path/to/calendar.ics
```

Output of the GET request to the time set item can be text/calendar:

```
curl --request GET --user administrator:administrator --header Accept: text/calendar http
\> /path/to/download/calendar.ics
```

or application/json:

```
curl --request GET --user administrator:administrator --header Accept: application/json
https://127.0.0.1:1443/api/timesets/12
```

By default API will send response in text/calendar format.

Output will be generated iCalendar including time set events:

```
curl --request GET --user administrator:administrator 'https://127.0.0.1:1443/api/timesets ←
/12'
BEGIN:VCALENDAR
PRODID:-//Mozilla.org/NONSGML Mozilla Calendar V1.1//EN
NAME:api_test_timeset_name2
VERSION:2.0

BEGIN:VEVENT
UID:sipwise19@sipwise15
SUMMARY:unique_name event 19
DTSTART:19710101T000001
DTEND:20201231T235959
END:VEVENT
END:VCALENDAR+
```

## 7.31 Generation of 181 Call Is Being Forwarded

### 7.31.1 Overview

SIP message *181 - Call Is Being Forwarded* is an optional provisional response that can be sent towards the caller user to inform it of an ongoing call forward. The message can also contain History-Info headers necessary to the caller to identify which is the new destination of the call.

### 7.31.2 How to enable it

By default Sipwise C5 doesn't generate any 181 message. To enable this feature two steps are required:

- set config.yml option *kamailio.proxy.cf\_send\_181.enable* to *yes*
- set config.yml option *sems.sbc.reset\_tag\_on\_fork* to *yes*

The first preference will actually activate the 181 message generation. The second one, instead, allows SEMS to forward back to caller all the provisional messages even if they contain a different To-Tag. Without this second preference set, the 181 message is sent towards the caller but all the following 18x messages are blocked by SEMS.

Additionally to add the History-Info headers to the 181 message:

- the option *outbound\_history\_info* has to be set in subscriber/domain preferences

### 7.31.3 How it works

When a call forward is triggered in Sipwise C5, Kamailio Proxy stores in the Redis database, which is selected by *kamailio.proxy.cf\_send\_181* preference in config.yml, the History-Info headers that are added to the outgoing INVITE message. Kamailio LB receives the outgoing INVITE and it generates a *182 - Connecting* message back towards the caller. The 182 message is received by Kamailio Proxy that converts it in a *181 - Call Is Being Forwarded* message and adds the History-Info headers previously stored in the Redis DB, if any. The message is then sent back to the caller.

## 8 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.

### 8.1 The Customer Self-Care Web Interface

The Sipwise C5 provides a web panel for end users (CSC panel) to maintain their subscriber accounts, which is running on <https://ngcp-ip>. Every subscriber can log in there, change subscriber feature settings, view their call lists, retrieve voicemail messages and trigger calls using the click-to-dial feature.

#### 8.1.1 Login Procedure

To log into the CSC panel, the end user has to provide his full web username (e.g. `user1@1.2.3.4`) and the web password defined in Section 6.3. Once logged in, he can change his web password in the *Account* section. This will NOT change his SIP password, so if you control the end user devices, you can auto-provision the SIP password into the device and keep it secret, and just hand over the web password to the customer. This way, the end user will only be able to place calls with this auto-provisioned device and not with an arbitrary soft-phone, but can nonetheless manage his account via the CSC panel.

#### 8.1.2 Site Customization

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

##### 8.1.2.1 Changing the Logo

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

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

The web panel customisation happens as follows:

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

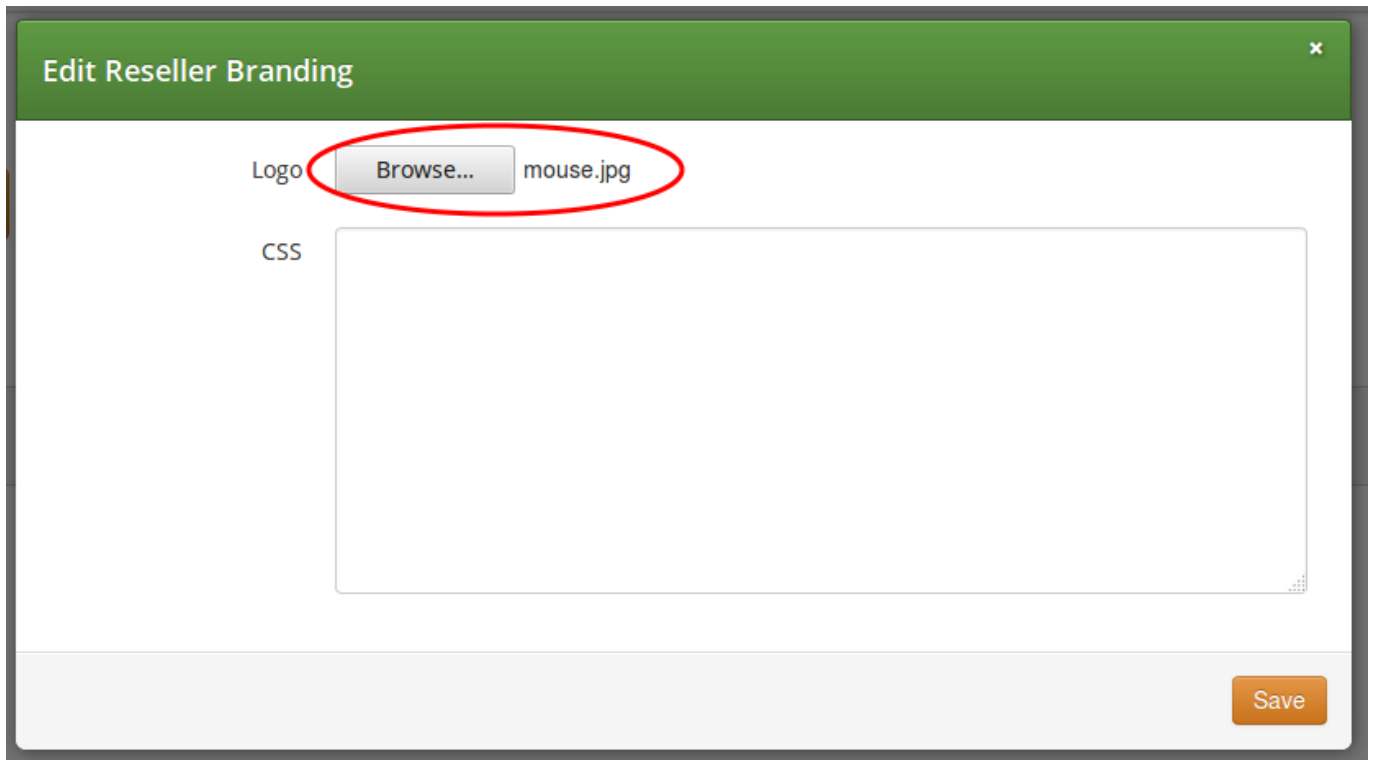



Figure 67: CSC Customisation Step 1: Select an image

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

Reseller branding successfully updated

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

### Custom Logo



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

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

### Custom CSS

Figure 68: CSC Customisation Step 2: Copy CSS code

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

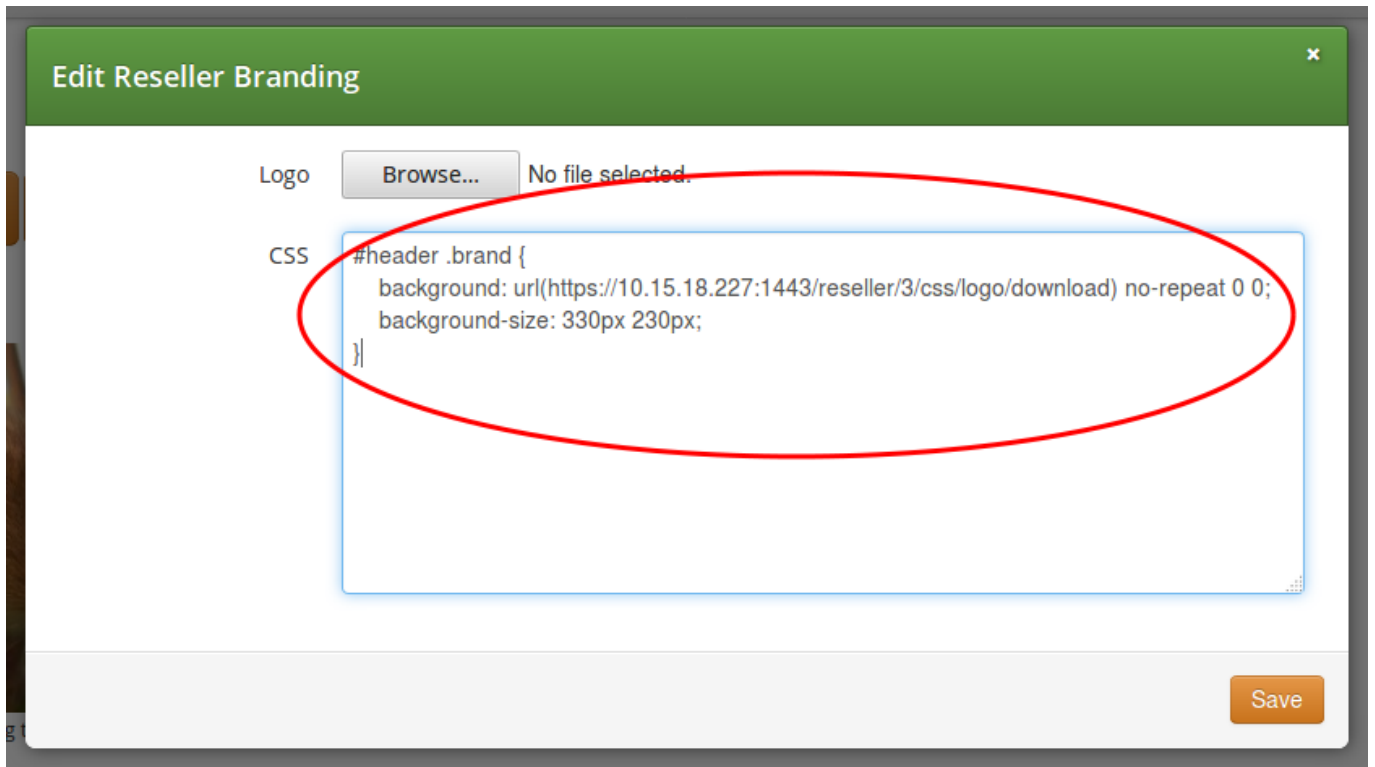


Figure 69: CSC Customisation Step 3: Paste CSS code

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



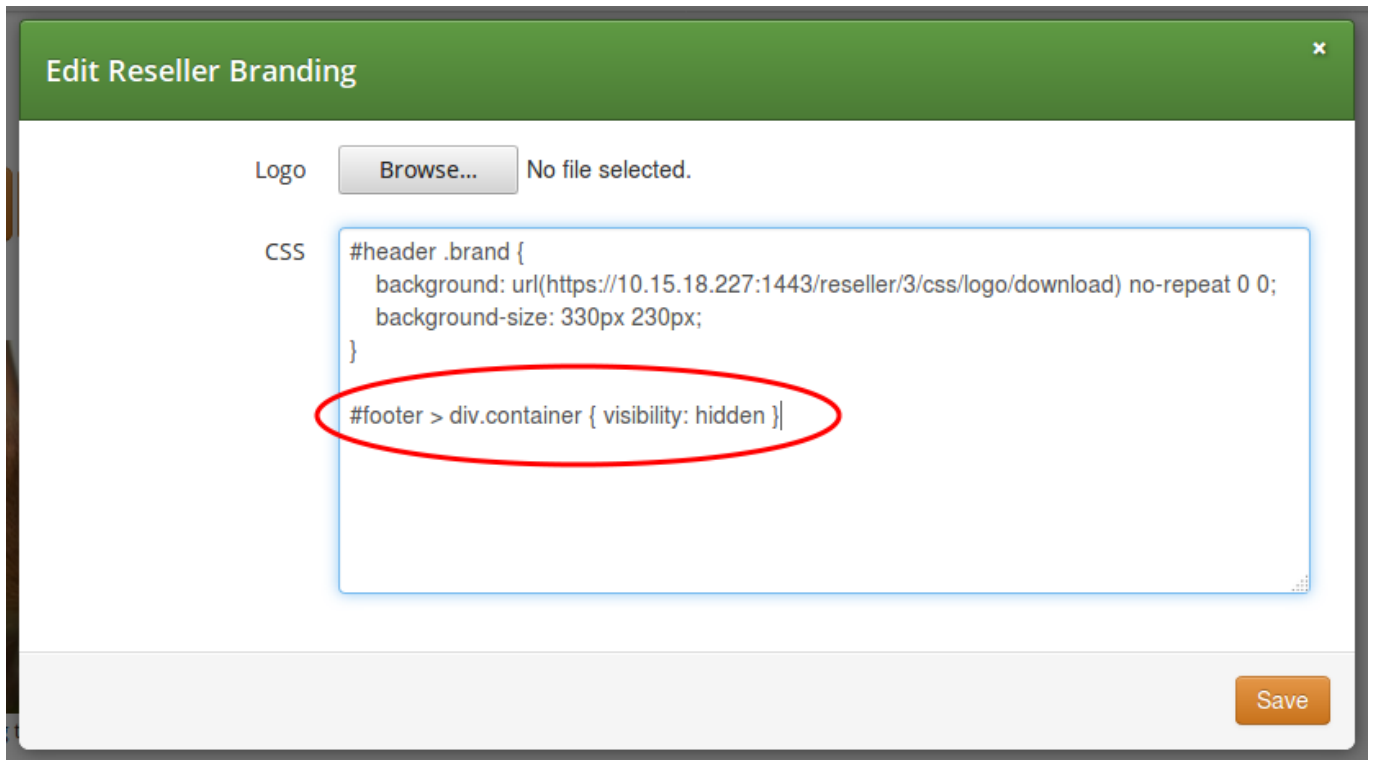


Figure 70: CSC Customisation: Hide copyright notice

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

Reseller branding successfully updated

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

Custom Logo



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

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

Custom CSS

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

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

Figure 71: CSC Customisation: Custom data on panel

### 8.1.2.2 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/`

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.*

### 8.1.2.3 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-co` file. An example to set the English language as default: `force_language: en`

## 8.2 The Voicemail Menu

Sipwise C5 offers several ways to access the Voicemail box.

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

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

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

---

### Tip

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

---

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

- As an operator, you can setup some DID number as external voicemail access number: for that, you should add a special rewrite rule (Inbound Rewrite Rule for Callee, see Section 6.7.) on the incoming peer, to rewrite that DID to "voiceboxpass". Now when user calls this number the call will be forwarded to the voicemail server and he will be prompted for mailbox and password. The mailbox is the full E.164 number of the subscriber account and the password is the PIN set in the CSC panel.
- The user can also dial his own number from PSTN, if he setup Call Forward on Not Available to the Voicebox, and when reaching the voicemail server he can interrupt the "user is unavailable" message by pressing \* key and then be prompted for the PIN. After entering PIN and confirming with # key he will enter own voicemail menu. PIN is random by default and must be kept secret for that reason.

## 9 Billing Configuration

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

### 9.1 Billing Profiles

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

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

#### 9.1.1 Creating Billing Profiles

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

Logged in as administrator Logout

### Create Billing Profiles

Reseller Search:

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

Showing 1 to 1 of 1 entries

← 1 →

Create Reseller

Handle  2

Name  3

Prepaid

Interval charge

4 Save

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

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

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

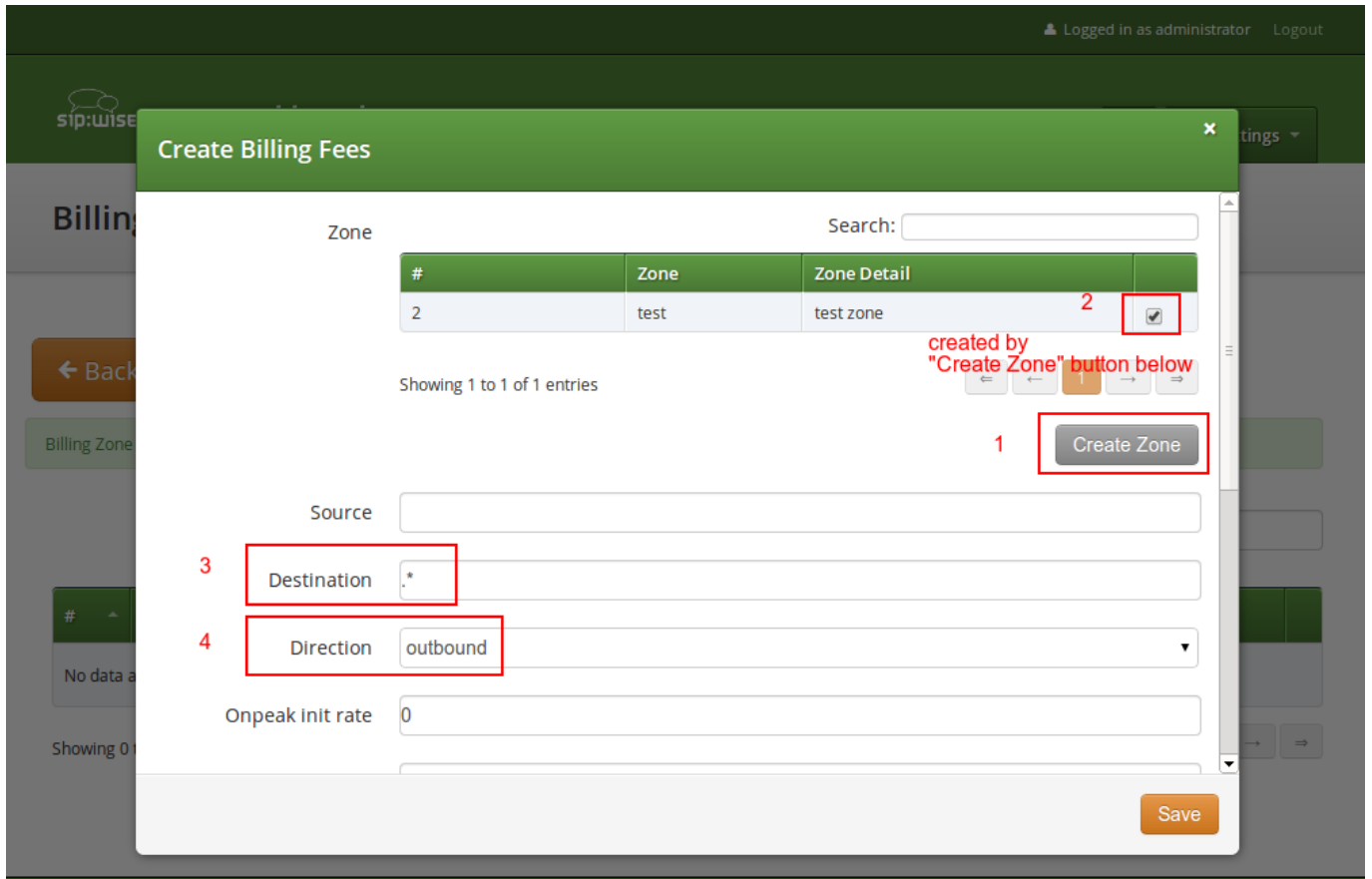
### 9.1.2 Creating Billing Fees

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

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

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

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



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

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

source/destination from the CDR. The fee with the longest (most distinctive) destination pattern is picked. If there are multiple, the one with the longest (most distinctive) source pattern is picked. In csv files, this match mode is specified by `regex_longest_pattern`.

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

- **Source:** The source pattern (prefix ie. 123 or regular expression `^123someone@sip\.sipwise\.com$`). The legacy default " ." regular expression (matching everything) will be set implicitly.
- **Destination:** The destination pattern (string ie. 456somebody@sip.sipwise.com, prefix ie. 456 or regular expression `^456somebody@sip\.sipwise\.com$`). This field must be set.
  - To specify a special fixed rate for any ported number in the local LNP tables belonging to an LNP provider, a fee with `exact_destination` match mode and `destination lnp:<lnp provider ID>` can be set up.
  - To specify an FCI (Furnished Charging Info) destination for cases when the FCI data is retrieved from the LNP lookup, use a format `fci=10050` where "10050" is the FCI data.
- **Direction:** `Outbound` for standard origination fees (applies to callers placing a call and getting billed for that) or `Inbound` for termination fees (applies to callees if you want to charge them for receiving various calls, e.g. for 800-numbers). *If in doubt, use Outbound.* If you upload fees via CSV files, use `out` or `in`, respectively.



#### Important

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

---



#### Important

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

---

- **Onpeak init rate:** The rate for the first rating interval in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during onpeak hours.
- **Onpeak init interval:** The duration of the first billing interval, in seconds. Applicable to calls during onpeak hours.
- **Onpeak follow rate:** The rate for subsequent rating intervals in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during onpeak hours. Defaults to *onpeak init rate*.
- **Onpeak follow interval:** The duration of subsequent billing intervals, in seconds. Applicable to calls during onpeak hours. Defaults to *onpeak init interval*.



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

### 9.1.3 Creating Off-Peak Times

To be able to differentiate between on-peak and off-peak calls, the platform stores off-peak times for every billing profile based on weekdays and/or calendar days. To edit the settings for a billing profile, go to *Settings*→*Billing* and press the *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.

The screenshot shows the 'Edit Monday' modal in the Sipwise interface. The modal contains three time input fields. The first field shows '00:00:00 - 07:59:59'. The second field, highlighted with a red box and labeled '1', contains '18:00:00'. The third field, highlighted with a red box and labeled '2', contains '23:59:59'. To the right of these fields is a trash icon and an 'Add' button, which is also highlighted with a red box and labeled '3'. Below the modal, a table lists weekdays and their corresponding start and end times.

| 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.

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

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

3 Save

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

## 9.2 Fraud Detection and Locking

The Sipwise C5 supports a fraud detection feature, which is designed to detect accounts causing unusually high customer costs, and then to perform one of several actions upon those accounts. This feature can be enabled and configured through two sets of billing profile options described in Section 9.1.1, namely the monthly (*fraud monthly limit*, *fraud monthly lock* and *fraud monthly notify*) and daily limits (*fraud daily limit*, *fraud daily lock* and *fraud daily notify*). Either monthly/daily limits or both of them can be active at the same time.

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

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

### 9.2.1 Fraud Lock Levels

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

The following lock levels are available:

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



#### Important

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

---



#### Caution

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

---



#### Important

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

---

#### Note

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

---

## 9.3 Notes on Billing and Call Rating

### *Cash balance with post-paid billing profile*

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

---

**Tip**

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

---

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

**Important**

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

---

**Note**

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

---

## 9.4 Billing Data Export

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

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

### 9.4.1 Glossary of Terms

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

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

- **Reseller:** the entity who is the direct, administrative service provider of a group of customers and subscribers registered on NGCP; Sipwise C5 operator may also charge a reseller for the calls initiated or received by its subscribers
- **User:** the subscriber who either is registered on NGCP, or is an external call party

#### 9.4.2 File Name Format

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

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

The definition of the specific fields is as follows:

Table 10: CDR/EDR export file name format

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

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

```
sipwise_007_20130310143000_0000000042.cdr
```

#### 9.4.3 File Format

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

##### 9.4.3.1 File Header Format

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

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

The definition of the specific fields is as follows:

Table 11: CDR/EDR export file header line format

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

A valid example for a Header is:

```
007,0738
```

#### 9.4.3.2 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/cdr` file. Each line holds one call detail record in CSV format and is constructed by a configurable set of fields, all of them enclosed in single quotes.

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

Table 12: Default set of system CDR fields

| Body Element       | Length | Type      | Description   |
|--------------------|--------|-----------|---|
| CDR_ID             | 1-10   | uint      | Internal CDR ID.  |
| UPDATE_TIME        | 19     | timestamp | Timestamp of last modification, including date and time (with seconds precision).   |
| SOURCE_USER_ID     | 36     | string    | Internal UUID of calling party subscriber. Value is 0 if calling party is external. |
| SOURCE_PROVIDER_ID | 0-255  | string    | Internal ID of the contract of calling party provider (i.e. reseller or peer).      |

Table 12: (continued)

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



Table 12: (continued)

| Body Element         | Length | Type                         | Description  |
|----------------------|--------|------------------------------|--|
| 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:<br>call: normal call<br>cfu: call forward unconditional<br>cfb: call forward busy<br>cft: call forward timeout<br>cfna: call forward not available<br>cfs: call forward for SMS<br>cfr: call forward on response<br>cfo: call forward on overflow       |
| CALL_STATUS          | 2-8    | string                       | The final call status - one of:<br>ok: successful call<br>busy: called party busy<br>noanswer: no answer from called party<br>cancel: cancel from caller<br>offline called party offline<br>timeout: no reply from called party<br>other: unspecified, see CALL_CODE field for details |
| CALL_CODE            | 3      | string                       | The final SIP status code.   |
| INIT_TIME            | 23     | timestamp                    | Timestamp of call initiation (SIP <i>INVITE</i> received from calling party). Includes date, time with milliseconds (3 decimals).  |
| START_TIME           | 23     | timestamp                    | Timestamp of call establishment (final SIP response received from called party). Includes date, time with milliseconds (3 decimals).   |
| DURATION             | 4-13   | fixed precision (3 decimals) | Length of call (calculated from START_TIME) including milliseconds (3 decimals).   |
| END_TIME             | 23     | timestamp                    | START_TIME plus DURATION.  |
| INIT_TIME_TRUNCATED  | 19     | timestamp                    | INIT_TIME without milliseconds.  |
| START_TIME_TRUNCATED | 19     | timestamp                    | START_TIME without milliseconds.   |
| END_TIME_TRUNCATED   | 19     | timestamp                    | END_TIME without milliseconds.   |

Table 12: (continued)

| Body Element                   | Length | Type                         | Description  |
|--------------------------------|--------|------------------------------|--|
| TIMEZONE                       | 100    | string                       | The name of the local subscriber's active timezone, defined by the contact of the subscriber/contract/reseller. The caller's timezone is used for outgoing calls. For calls from external subscribers, the callee's timezone is used. System timezone (defined in config.yml) is used for transit calls. |
| INIT_TIME_LOCALIZED            | 23     | timestamp                    | INIT_TIME, converted to TIMEZONE.  |
| START_TIME_LOCALIZED           | 23     | timestamp                    | START_TIME, converted to TIMEZONE.   |
| END_TIME_LOCALIZED             | 23     | timestamp                    | END_TIME, converted to TIMEZONE.   |
| INIT_TIME_LOCALIZED_TRUNCATED  | 19     | timestamp                    | INIT_TIME_LOCALIZED without milliseconds.  |
| START_TIME_LOCALIZED_TRUNCATED | 19     | timestamp                    | START_TIME_LOCALIZED without milliseconds.   |
| END_TIME_LOCALIZED_TRUNCATED   | 19     | timestamp                    | END_TIME_LOCALIZED without milliseconds.   |
| CALL_ID                        | 0-255  | string                       | The SIP Call-ID.   |
| RATING_STATUS                  | 2-7    | string                       | The internal rating status of the CDR - one of:<br>unrated: not rated<br>ok: successfully rated<br>failed: error while rating<br>Currently always ok or unrated, depending on whether rating is enabled or not.  |
| RATED_AT                       | 0-19   | datetime                     | Time of rating, including date and time (with seconds precision). Empty if CDR is not rated.   |
| SOURCE_CARRIER_COST            | 7-14   | fixed precision (6 decimals) | The originating carrier cost that the carrier (i.e. SIP peer) charges for the calls routed to his network, or empty if CDR is not rated.<br><i>PLEASE NOTE: Only available in system exports, not for resellers.</i>   |
| SOURCE_CUSTOMER_COST           | 7-14   | fixed precision (6 decimals) | The originating customer cost, or empty if CDR is not rated.   |

Table 12: (continued)

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

Table 12: (continued)

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

Table 12: (continued)

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

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

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

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

---

#### Note

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

---

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

#### 9.4.3.3 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 Sipwise C5 for CDR processing but rather provides the system administrator with a possibility to include supplementary information in CDRs.

---

**Note**

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

---

Table 13: Supplementary data in CDR fields

| Body Element     | Length | Type   | Description                                  |
|------------------|--------|--------|--|
| SOURCE_GPP0      | 0-255  | string | Supplementary data field 0 of calling party. |
| SOURCE_GPP1      | 0-255  | string | Supplementary data field 1 of calling party. |
| SOURCE_GPP2      | 0-255  | string | Supplementary data field 2 of calling party. |
| SOURCE_GPP3      | 0-255  | string | Supplementary data field 3 of calling party. |
| SOURCE_GPP4      | 0-255  | string | Supplementary data field 4 of calling party. |
| SOURCE_GPP5      | 0-255  | string | Supplementary data field 5 of calling party. |
| SOURCE_GPP6      | 0-255  | string | Supplementary data field 6 of calling party. |
| SOURCE_GPP7      | 0-255  | string | Supplementary data field 7 of calling party. |
| SOURCE_GPP8      | 0-255  | string | Supplementary data field 8 of calling party. |
| SOURCE_GPP9      | 0-255  | string | Supplementary data field 9 of calling party. |
| DESTINATION_GPP0 | 0-255  | string | Supplementary data field 0 of called party.  |
| DESTINATION_GPP1 | 0-255  | string | Supplementary data field 1 of called party.  |
| DESTINATION_GPP2 | 0-255  | string | Supplementary data field 2 of called party.  |
| DESTINATION_GPP3 | 0-255  | string | Supplementary data field 3 of called party.  |
| DESTINATION_GPP4 | 0-255  | string | Supplementary data field 4 of called party.  |
| DESTINATION_GPP5 | 0-255  | string | Supplementary data field 5 of called party.  |
| DESTINATION_GPP6 | 0-255  | string | Supplementary data field 6 of called party.  |
| DESTINATION_GPP7 | 0-255  | string | Supplementary data field 7 of called party.  |
| DESTINATION_GPP8 | 0-255  | string | Supplementary data field 8 of called party.  |
| DESTINATION_GPP9 | 0-255  | string | Supplementary data field 9 of called party.  |

**Account balance details (prepaid calls)**

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

---

**Tip**

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

---

**Note**

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

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

1. `source|destination`: decides if the data refers to calling (`source`) or called (`destination`) party
2. `carrier|reseller|customer`: the account owner, whose billing data is referred
3. data type:
  - 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.

#### 9.4.3.4 Distinguish between on-net and off-net calls CDRs

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

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

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

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

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

Table 14: Default set of system EDR fields

| Body Element           | Length | Type   | Description   |
|------------------------|--------|--------|---|
| EVENT_ID               | 1-11   | uint   | Internal EDR ID.  |
| TYPE                   | 0-255  | string | The type of the event - one of:<br>start_profile: A subscriber profile has been newly assigned to a subscriber.<br>end_profile: A subscriber profile has been removed from a subscriber.<br>update_profile: A subscriber profile has been changed for a subscriber.<br>start_huntgroup: A subscriber has been provisioned as PBX / hunting group.<br>end_huntgroup: A subscriber has been deprovisioned as PBX / hunting group.<br>start_ivr: A subscriber has a new call-forward to Auto-Attendant.<br>end_ivr: A subscriber has removed a call-forward to Auto-Attendant. |
| CONTRACT_EXTERNAL_ID   | 0-255  | string | The external ID of the customer. (A string value shown as "External ID" property of an Sipwise C5 customer.)  |
| COMPANY                | 0-127  | string | The company name of the customer's contact.   |
| SUBSCRIBER_EXTERNAL_ID | 0-255  | string | The external ID of the subscriber. (A string value shown as "External ID" property of an Sipwise C5 subscriber.)<br><i>PLEASE NOTE: This field is empty in case of start_huntgroup and end_huntgroup events.</i>  |
| PILOT_PRIMARY_NUMBER   | 0-64   | string | The pilot subscriber's primary number (HPBX subscribers). <i>PLEASE NOTE: This is not included in default set of EDR fields from Sipwise C5 version mr5.0 upwards.</i>  |
| PRIMARY_NUMBER         | 0-64   | string | The VoIP number of the subscriber with the highest ID (DID or primary number).  |
| OLD_PROFILE_NAME       | 0-255  | string | The old status of the event. Depending on the event_type:<br>start_profile: Empty.<br>end_profile: The name of the subscriber profile which got removed from the subscriber.<br>update_profile: The name of the former subscriber profile which got updated.<br>start_huntgroup: Empty.<br>end_huntgroup: Empty.<br>start_ivr: Empty.<br>end_ivr: Empty.  |



Table 14: (continued)

| Body Element      | Length | Type      | Description  |
|-------------------|--------|-----------|--|
| NEW_PROFILE_NAME  | 0-255  | string    | The new status of the event. Depending on the event_type:<br>start_profile: The name of the subscriber profile which got assigned to the subscriber.<br>end_profile: Empty.<br>update_profile: The name of the new subscriber profile which got applied.<br>start_huntgroup: Empty.<br>end_huntgroup: Empty.<br>start_ivr: Empty.<br>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.<br><i>PLEASE NOTE: Only available in system exports, not for resellers.</i>  |
| <line_terminator> | 1      | string    | A fixed character. Always \n (special char LF - ASCII 0x0A).   |

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

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

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

---

#### Note

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

---

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

**9.4.3.6 Extra fields that can be exported to EDRs**

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

Table 15: Supplementary data in EDR fields

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

### 9.4.3.7 File Trailer Format

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

```
<md5 sum>
```

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

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

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

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

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

```
$ ngcp-cdr-md5 sipwise_001_20071110123000_0000000004.cdr
/tmp/ngcp-cdr-md5.sipwise_001_20071110123000_0000000004.cdr.hUtuhtKE1: FAILED
md5sum: WARNING: 1 of 1 computed checksum did NOT match
```

### 9.4.4 File Transfer

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

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

If public key authentication is chosen, the public key file has to be stored in the file `~/.ssh/authorized_keys2` below the home directory of the `cdrexp` user. Otherwise, a password has to be set for the user.

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

---

#### Note

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

---

## 10 Corosync/Pacemaker

### 10.1 Overview

The Corosync/Pacemaker pair is the successor of the long obsolete and unsupported Heartbeat v2 software package. While Heartbeat v2 was playing the role of both the **Group Communication System** (GCS) and the **Cluster Resource Manager** (CRM), these roles are split under the new system. Corosync is the GCS and in charge of communication, while Pacemaker sits on top of the GCS and plays the role of the CRM, managing the resources and responding to changes in the cluster status.

### 10.2 Migration

Before the migration install the latest hotfixes for the packages: `ngcp-system-tools-pro`, `ngcp-ngcpcfg`, `ngcp-ngcpcfg-ha`.

An existing cluster under Heartbeat v2 can be migrated to Corosync/Pacemaker using the script `ngcp-migrate-ha-crm`. This script automates the following steps:

1. Locally download the Heartbeat v2 Debian package to allow rollback
2. Download and prefetch the Debian packages for Corosync, Pacemaker, and its dependencies
3. Remove `/etc/ha.d/haresources` to disable stopping of resources by Heartbeat v2
4. Stop Heartbeat v2 and remove its Debian package including all configuration
5. Install Corosync, Pacemaker, and all dependencies
6. Switch `config.yml` variables `ha.gcs` and `ha.crm` to `corosync` and `pacemaker` respectively
7. Rebuild all config from templates
8. Start the new GCS and CRM services

The script must be run on the standby node, and the steps described above are first performed locally (on the standby node), followed by the node's peer (the active node). This is to minimise resource downtime.



#### Caution

Switching the values of `ha.gcs` or `ha.crm` is not enough to actually effect any change. These values are merely reflective of which software is installed and must not be modified without also altering the underlying software.

---

The migration needs to be run on non-migrated pairs of nodes, from the standby node of each pair, and the services need to be all in a good state (from `ngcp-service summary` point of view). The program will perform these sanity checks before making it possible to proceed. The user will also be prompted for confirmation, which can be skipped for non-interactive use with the `FORCE` environment variable.

For a Carrier system the configuration settings will be set per pair, so that it does not affect the entire cluster. Once the whole cluster has been migrated these configuration files should be merged into the global one. If the switch does not need to be staged, then the

`ngcp-parallel-ssh(8)` command can help with that, with its inactive host selector, such as `ngcp-parallel-ssh inactive "FORCE=yes ngcp-migrate-ha-crm"`.

### 10.2.1 Rollback

Rollback can be done by using `ngcp-migrate-ha-crm --rollback`, which involves reversing the steps outlined above: Removal of `corosync` and `pacemaker`, installation of the downloaded `heartbeat-2` Debian package, and reverting `ha.gcs` and `ha.crm` to their previous values `heartbeat-2`.

## 10.3 Corosync

Corosync is the **Group Communication System** (GCS). Its configuration resides in `/etc/corosync/corosync.conf` and describes the following details:

- Shared cluster name of `sp`
- Quorum config as a two-node cluster (see below)
- Config details for both nodes:
  - Name (`sp1` and `sp2`, or `a` and `b` node names)
  - Node ID (1 and 2 respectively)
  - Local IP address for communication

### 10.3.1 Quorum

Corosync uses a voting system to determine the state of the cluster. Each configured node in the cluster receives one vote. A quorum is defined as a majority presence within the cluster, meaning at least 50% of the configured nodes plus one, or  $q = n / 2 + 1$ . For example, if 8 nodes were configured, a quorum would be present if at least 5 nodes are communicating with each other. In this state, the cluster is said to be quorate, which means it can operate normally. (Any remaining nodes, 3 in the worst case, would see the cluster as inquorate and would relinquish all their resources.)

A two-node cluster is a special case as under the formula above, a quorum would consist of 2 functioning nodes. The Corosync config setting `two_node: 1` overrides this and artificially sets the quorum to 1. This means that under a split-brain scenario (each node seeing only 1 vote), both nodes would see the cluster as quorate and try to become active, instead of both nodes going standby.

In addition to this, Pacemaker itself also uses an internal scoring system for individual resources. This mechanism is described below and not directly related to the quorum.

## 10.4 Pacemaker

Pacemaker uses the communication service provided by Corosync to manage local resources. All status and configuration information is shared between all Pacemaker instances within the cluster as long as communication is up. This means that any

configuration change done on any node will immediately and automatically be propagated to all other nodes in the cluster.

Pacemaker internally uses an XML document to store its configuration, called "CIB" stored in `/var/lib/pacemaker/cib/cib.xml`. However, this XML document **must never** be edited or modified directly. Instead, a shell-like interface `crm` is provided to talk to Pacemaker, query status information, alter cluster state, view and modify configuration, etc. Any configuration change done through `crm` is immediately reflected in the CIB XML, locally as well as on all other nodes.

**Warning**

To repeat, do not ever directly modify Pacemaker's XML configuration.

---

As an added bonus, just to make things more awkward, the syntax used by `crm` is not XML at all, but rather uses a Cisco-like hierarchy.

Commands can be issued to `crm` either directly from the shell as command-line arguments, or interactively by entering a Cisco-like shell. So for example, the current config can be viewed either from the shell with:

```
root@spl:~# crm config show
...
```

Or interactively, as either:

```
root@spl:~# crm
crm(live/spl)# config
crm(live/spl)configure# show
...
```

or

```
root@spl:~# crm
crm(live/spl)# config show
...
```

Interactive online help is provided by the `ls` command to list commands valid in the current context, or using the `help` command for a more verbose help output.

## 10.5 Query Status

The current cluster status can be viewed with the top-level `status` command:

```

crm(live/sp1)# status
Stack: corosync
Current DC: sp2 (version unknown) - partition with quorum
Last updated: Fri Nov 22 18:38:06 2019
Last change: Fri Nov 22 18:25:28 2019 by hacluster via crmd on sp1

```

```

2 nodes configured
7 resources configured

```

```

Online: [ sp1 sp2 ]

```

Full list of resources:

```

Resource Group: g_vips
  p_vip_eth1_v4_1 (ocf::heartbeat:IPaddr):      Started sp1
  p_vip_eth2_v4_1 (ocf::heartbeat:IPaddr):      Started sp1
Resource Group: g_ngcp
  p_monit_services (ocf::ngcp:monit-services):  Started sp1
Clone Set: c_ping [p_ping]
  Started: [ sp1 sp2 ]
Clone Set: fencing [st-null]
  Started: [ sp1 sp2 ]

```

If the status is queried from `sp2` instead, the output will be the same. Most importantly, the resources will **not** show up as "stopped" on `sp2` but instead will be reported as running on `sp1`.

The resources reported are described in the configuration section below.

## 10.6 Config Management

The NGCP templates do not operate on Pacemaker's CIB XML directly, but instead produce a file in CRM syntax in `/etc/pacemaker/`. This file is not handled by Pacemaker directly, but instead is loaded into Pacemaker via the `crm` command `config load replace`. It shouldn't be necessary to do this manually, as the script `ngcp-ha-crm-reload` handles this automatically, which is called from the config file's postbuild script.

Changes to the config don't need to be saved explicitly. This is done automatically by Pacemaker, as well as sharing any changes with all other members of the cluster.

In `crm`, changes made to the config are cached until made active with `commit`, or discarded with `refresh`. Changes to resource status can be avoided by enabling maintenance mode (see below).



### Important

However, since our config is loaded from a template, any changes done to the config through `crm` manually are transient and will be lost the next time a config reload happens.

---

The currently active config can be shown with `config show` and should be logically identical to the contents of `/etc/pacemaker/c`

```
crm(live/sp1)# config show
node 1: sp1
node 2: sp2
primitive p_monit_services ocf:ngcp:monit-services \
    meta migration-threshold=20 \
    meta failure-timeout=800 \
    op monitor interval=20 timeout=60 on-fail=restart \
    op_params on-fail=restart
primitive p_ping ocf:pacemaker:ping \
    params host_list="10.15.20.30 192.168.211.1" multiplier=1000 dampen=5s \
    meta failure-timeout=800 \
    op monitor interval=1 timeout=60 on-fail=restart \
    op_params timeout=60 on-fail=restart
primitive p_vip_eth1_v4_1 IPaddr \
    params ip=192.168.255.250 nic=eth1 cidr_netmask=24 \
    op monitor interval=5 timeout=60 on-fail=restart \
    op_params on-fail=restart
primitive p_vip_eth2_v4_1 IPaddr \
    params ip=192.168.1.161 nic=eth2 cidr_netmask=24 \
    op monitor interval=5 timeout=60 on-fail=restart \
    op_params on-fail=restart
primitive st-null stonith:null \
    params hostlist="sp1 sp2"
group g_ngcp p_monit_services
group g_vips p_vip_eth1_v4_1 p_vip_eth2_v4_1
clone c_ping p_ping
clone fencing st-null
location l_ngcp g_ngcp \
    rule pingd: defined pingd
colocation l_ngcp_with_vip inf: g_ngcp g_vips
location l_vips g_vips \
    rule pingd: defined pingd
order o_vip_then_ngcp Mandatory: g_vips g_ngcp
```



```
property cib-bootstrap-options: \
    have-watchdog=false \
    cluster-infrastructure=corosync \
    cluster-name=sp \
    stonith-enabled=yes \
    no-quorum-policy=ignore \
    startup-fencing=yes \
    maintenance-mode=false \
    last-lrm-refresh=1574443528
rsc_defaults rsc-options: \
    resource-stickiness=100
```

### 10.6.1 General Concepts

- The configuration consists of a collection of objects of various types with various attributes.
- Each object has a unique identifying name that can be used to refer to it.
- Usually the type of the object is the first word and the identifying name is the second. For example, `clone c_ping p_ping` defines a `clone` type object with the name `c_ping`.
- The unique name is used e.g. when deleting an object (`config del ...`), when starting or stopping a resource, when referring to resources from a group, etc.

### 10.6.2 Resources

Resources are the primary type of objects that Pacemaker handles. A resource is anything that can be started or stopped, and a resource is normally allowed to run on one node only. A resource is defined as a `primitive` type object.

Pacemaker supports many types of resources, all of which have different options that can be given to them. The config syntax defines that options given to a resource itself are prefixed with `params`, while options that influence how a resource should be managed are prefixed with `meta`. Options that are relevant to operations that can be performed on a resource are prefixed with `op`.

Resources are grouped into classes, providers, and types. Details about them (e.g. which options they support) can be obtained through the `ra` menu.

```
crm(live/sp1)ra# info IPaddr
Manages virtual IPv4 addresses (portable version) (ocf:heartbeat:IPaddr)
...
```

#### 10.6.2.1 Shared IP Addresses

```
primitive p_vip_eth1_v4_1 IPAddr \
  params ip=192.168.255.250 nic=eth1 cidr_netmask=24 \
  op monitor interval=5 timeout=60 on-fail=restart \
  op_params on-fail=restart
```

This defines a resource of type `IPAddr` with name `p_vip_eth1_v4_1` and the given parameters (address, netmask, interface). Pacemaker will check for the existence of the address every 5 seconds, with an action timeout of 60 seconds. If the monitor action fails, the resource is restarted.

### 10.6.2.2 System Services

```
primitive p_monit_services ocf:ngcp:monit-services \
  meta migration-threshold=20 \
  meta failure-timeout=800 \
  op monitor interval=20 timeout=60 on-fail=restart \
  op_params on-fail=restart
```

While Pacemaker has support for native `systemd` services, for the time being we're still relying on `monit` to manage our services. Therefore, services are defined in Pacemaker virtually identical to how they were defined in Heartbeat v2, through a `monit-services start/stop script`. The old Heartbeat script was `/etc/ha.d/resource.d/monit-services` and the new script used by Pacemaker is `/etc/ngcp-ocf/monit-services`.

---

#### Note

The primary difference between the two scripts is the support for a `monitor` action for Pacemaker, which can be configured via the `config.yml` variable `ha.monitor_services`. It can be set to *full* to periodically use the output of `ngcp-service summary` to determine whether all services are running or not. The default value is *none*, which preserves backwards compatibility with the behavior of Heartbeat v2, by performing no periodic checks of the status of the services.

---

- `meta migration-threshold=20` means that the resource will be migrated away (instead of restarted) after 20 failures. See the discussion on failure counts below.
- `meta failure-timeout=800` means that the failure count should be reset to zero if the last failure occurred more than 800 seconds ago. (However, the actual timer depends on the `cluster-recheck-interval`.)
- Run the monitor action every 20 seconds with a timeout of 60 seconds and restart the resource on failure.

### 10.6.2.3 Ping Nodes

```
primitive p_ping ocf:pacemaker:ping \
  params host_list="10.15.20.30 192.168.211.1" multiplier=1000 dampen=5s \
  meta failure-timeout=800 \
  op monitor interval=1 timeout=60 on-fail=restart \
```

```
op_params timeout=60 on-fail=restart
```

The builtin `pingd` service, using a resource name that intelligently is not named `pingd` but rather just `ping`, replaces Heartbeat's ping nodes. It supports multiple ping backends, and uses `fping` by default.

Each configured ping node (each entry in `host_list`) produces a score of 1 if that ping node is up. The scores are summed up and multiplied by the `multiplier`. So in the example above, a score of 2000 is generated if both ping nodes are up. Pacemaker will then prefer the node which produces the higher score.

- `dampen=5s` means to wait 5 seconds after a change occurred to prevent transient glitches from causing service flapping.
- Other options are the same as described above.

#### 10.6.2.4 Fencing/STONITH

```
primitive st-null stonith:null \
    params hostlist="sp1 sp2"
```

Pacemaker will generate a warning if no fencing mechanism is configured, therefore we configure the `null` fencing mechanism.

Pacemaker supports several proper fencing mechanism and these might eventually get supported in the future.

#### 10.6.2.5 Groups

```
group g_ngcp p_monit_services
group g_vips p_vip_eth1_v4_1 p_vip_eth2_v4_1
```

To manage, control, and restrict multiple resources at the same time, resources can be grouped into single objects. The group `g_ngcp` is pointless for the time being (it contains only a single other resource) but will become useful once native `systemd` resources are in use. The group `g_vips` ensures that all shared IP addresses are active at the same time.

#### 10.6.2.6 Clones

```
clone c_ping p_ping
clone fencing st-null
```

Since a single resource normally only runs on one node, a clone can be defined to allow a resource to run on all nodes. We want the `pingd` service and the fencing service to always run on all nodes.

#### 10.6.2.7 Constraints

```
colocation l_ngcp_with_vip inf: g_ngcp g_vips
```

This tells Pacemaker that we want to force the `g_ngcp` resource on the same node that is running the `g_vips` resource.

```
location l_ngcp g_ngcp \  
    rule pingd: defined pingd  
location l_vips g_vips \  
    rule pingd: defined pingd
```

This tells Pacemaker that these resources depend on the `pingd` service being healthy. If `pingd` fails on one node (ping nodes are unavailable), then Pacemaker will shut down the constrained resources.

```
order o_vip_then_ngcp Mandatory: g_vips g_ngcp
```

This tells Pacemaker that the shared IP addresses must be up and running before the system services can be started.

### 10.6.3 Cluster Options

```
property cib-bootstrap-options: \  
    have-watchdog=false \  
    cluster-infrastructure=corosync \  
    cluster-name=sp \  
    stonith-enabled=yes \  
    no-quorum-policy=ignore \  
    startup-fencing=yes \  
    maintenance-mode=false \  
    last-lrm-refresh=1574443528
```

Relevant options are:

- `have-watchdog=false` indicates that no external watchdog service such as SBD is in use.
- `cluster-name=sp` is to match the configuration of Corosync.
- `stonith-enabled=yes` is required to suppress a warning message, even though no real STONITH (null fencing mechanism) is in use.
- `no-quorum-policy=ignore` tells Pacemaker to continue normally if quorum is lost. This is the only setting that makes sense in a two-node cluster.
- `startup-fencing=yes` is also needed to suppress a warning even though no real fencing is in use. This tells Pacemaker to shoot nodes that are not present immediately after startup.
- `maintenance-mode=false` tells Pacemaker to actually perform resource actions. If maintenance mode is enabled, Pacemaker will continue to run, but will not start or stop any services. This should be enabled before loading a new config, and then disabled afterwards. The script `ngcp-ha-crm-reload` does this.

## 10.6.4 Failure Counts

Pacemaker keeps a failure count for each resource, which is somewhat hidden from view, but can largely influence its behaviour. Each time a service fails (either during runtime or during startup), the failure count is increased by one. If the failure count exceeds the configured `migration-threshold`, Pacemaker will cease trying to start the service and will migrate the service away to another node. In `crm status` this simply shows up as `stopped`.

Failure counts can be cleared automatically if the `failure-timeout` setting is configured for a resource. This timeout is counted after the last time the resource has failed, and is checked periodically according to the `cluster-recheck-interval`. In other words, a very short failure timeout won't have any effect unless the recheck interval is also very short.



### Important

If no failure timeout is configured, any existing failure count must be cleared manually.

---

### 10.6.4.1 Checking Failure Counts

The failure count for a resource can be checked from the shell via `crm_failcount`, for example:

```
root@sp1:~# crm_failcount -G -r p_monit_services
scope=status name=fail-count-p_monit_services value=0
```

The failure count on a different node can also be examined:

```
root@sp1:~# crm_failcount -G -r g_ngcp -N sp2
scope=status name=fail-count-g_ngcp value=0
```

The same can be done via `crm`:

```
crm(live/sp1)resource# failcount g_vips show sp1
scope=status name=fail-count-g_vips value=0
crm(live/sp1)resource# failcount c_ping show sp2
scope=status name=fail-count-c_ping value=0
```

As a shortcut, the script `ngcp-ha-show-failcounts` is provided:

```
root@sp1:~# ngcp-ha-show-failcounts
p_vip_eth1_v4_1:
  sp1: 0
```

```
sp2: 0
p_vip_eth2_v4_1:
sp1: 0
sp2: 0
p_monit_services:
sp1: 0
sp2: 0
```

#### 10.6.4.2 Clearing Failure Counts

Analogous to checking a failure count, it can be cleared using any one of these methods:

```
root@sp1:~# crm_failcount -D -r p_monit_services
Cleaned up p_monit_services on sp1
root@sp1:~# crm resource failcount g_ngcp delete sp2
Cleaned up p_monit_services on sp2
root@sp1:~# crm
crm(live/sp1)# resource
crm(live/sp1)resource# failcount c_ping delete sp1
Cleaned up p_ping:0 on sp1
Cleaned up p_ping:1 on sp1
crm(live/sp1)resource# bye
root@sp1:~# ngcp-ha-clear-failcounts
Cleaned up p_monit_services on sp2
Cleaned up p_monit_services on sp1
```

In addition, the `crm` command `resource cleanup` also resets failure counts.

#### 10.6.5 Resource Scores

Pacemaker uses an internal scoring system to determine which resources to run where. A resource will be run on the node on which it received the highest score. If a resource has a negative score, that resource will not be run at all. If a resource has the same score on multiple nodes, then the resource will be run on any one of those nodes. Scores can be calculated and acted upon through various config settings.

A score value of `infinity` (and negative infinity) to force certain states is provided, which evaluates to not infinity at all, but rather to a static value of one million. This can be used to artificially manipulate resource scores to force running a resource on a particular node, or forbid a resource from running on particular nodes.

Scores can be inspected through the `crm` command `resource scores`.

## 10.7 Common Tasks

### 10.7.1 Takeover and Standby

The commands `ngcp-make-active` and `ngcp-make-standby` work normally. Under Pacemaker, they function through the `crm` command `resource move` to create a temporary location constraint on `g_vips`. This can be done manually through:

```
crm resource move g_vips sp1 30
```

The lifetime of 30 seconds is needed because `g_ngcp` depends on the location of `g_vips`, and therefore `g_ngcp` needs to be stopped before `g_vips` can be stopped. The location constraint must remain active until `g_ngcp` has been completely and successfully stopped.

---

**Note**

These commands only effect the status of the running resources, and not the status of the node itself. This means that after going standby, Pacemaker will immediately be ready to take over the resources again if needed. See below for a discussion on node status.

---

Similarly, `ngcp-check-active` uses the output of `crm resource status g_vips` to determine whether the local node is active or not.

### 10.7.2 Node Status (Online/Standby)

In addition to the status and location of individual resources, nodes themselves can also go into standby mode. The submenu `node` in `crm` has the relevant options.

A node in standby mode will not only give up all of its resources, but will also refuse to take them over until it's back online. Therefore, it's possible to set both nodes to standby mode and shut down all resources on both nodes.

---

**Note**

A node in standby mode will still participate in GCS communications and remain visible to the rest of the cluster.

---

Use `crm node standby` to set the local node to standby mode. A remote node can be set to standby using e.g. `crm node standby sp2`.

By default, the standby mode remains active until it's cancelled manually (a lifetime of `forever`). Alternatively, a lifetime of `reboot` can be specified to tell Pacemaker that after the next reboot, the node should automatically come back online. Example:  
`crm node standby sp2 reboot`

To cancel standby mode, use `crm node online`, optionally followed by the node name.

To show the current status of all nodes, use `crm node show`. The top-level `crm status` also shows this.

### 10.7.3 Maintenance Mode

If Pacemaker's maintenance mode is enabled, it will continue to operate normally, i.e. continue to run and monitor resources, but will refuse to stop or start any resources. This is useful to make changes to the running config, and is done automatically by `ngcp-ha-crm-reload`.

To enable and disable maintenance mode:

```
crm maintenance on
crm maintenance off
```

or using the more lower level method:

```
crm configure property maintenance-mode=true
crm configure property maintenance-mode=false
```

### 10.7.4 CLI Alternatives

Several of the commands available through `crm` are also available through standalone CLI tools, such as `crm_failcount`, `crm_standby`, `crm_resource`, etc. They generally have less friendly syntax and so researching them is left as an exercise to the reader.



## 11 Provisioning REST API Interface

The Sipwise C5 provides the REST API interface for interconnection with 3rd party tools.

The Sipwise C5 provides a REST API to provision various functionality of the platform. The entry point - and at the same time the official documentation - is at <https://<your-ip>:1443/api>. It allows both administrators and resellers (in a limited scope) to manage the system.

You can either authenticate via username and password of your administrative account you're using to access the admin panel, or via SSL client certificates. Find out more about client certificate authentication in the online API documentation.

### 11.1 API Workflows for Customer and Subscriber Management

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

The standard life cycle of a customer and subscriber is:

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

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

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

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

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

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

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

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

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



### Important

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



### Warning

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

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

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

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

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

```

}
my $subscriber_id = $res->header('Location');
$subscriber_id =~ s/^.+\/(\d+)$/$1/; # extract the ID from the Location header
say "Created subscriber, id is $subscriber_id";

```



### Important

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

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

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

```

$req = HTTP::Request->new('PUT', "$uri/api/callforwards/$subscriber_id");
$req->header('Content-Type' => 'application/json');
$req->header('Prefer' => "return=minimal"); # use return=representation to get full json ←
      response
$req->content(JSON::to_json({
  cfna => { # set a call-forward if subscriber is not registered
    destinations => [
      { destination => "4366610001", timeout => 10 }, # ring this for 10s
      { destination => "4366710001", timeout => 300 }, # if no answer, ring that for ←
        300s
    ],
    times => undef # no time-based call-forward, trigger cfna always
  }
}));
$res = $ua->request($req);
if($res->code != 204) { # if return=representation, it's 200
  die "Failed to set cfna for subscriber: ".$res->decoded_content."\n";
}

```

You can set cfu, cfna, cfb, cft, cfs, cfr and cfo via this API call, also all at once. Destinations can be hunting lists as described above or just a single number. Also, a time set can be provided to trigger call forwards only during specific time periods.

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

```

$req = HTTP::Request->new('GET', "$uri/api/subscriberpreferences/$subscriber_id");
$res = $ua->request($req);
if($res->code != 200) {
  die "Failed to fetch subscriber preferences: ".$res->decoded_content."\n";
}

```

```

}
my $prefs = JSON::from_json($res->decoded_content);
delete $prefs->{_links}; # not needed in update

$prefs->{prepaid_library} = 'libinewrate'; # switch to inew billing
$prefs->{block_in_clir} = JSON::true; # reject incoming anonymous calls
$prefs->{block_in_list} = [ # reject calls from the following numbers:
    '4366412345', # this particular number
    '431*', # all vienna/austria numbers
];
$req = HTTP::Request->new('PUT', "$uri/api/subscriberpreferences/$subscriber_id");
$req->header('Content-Type' => 'application/json');
$req->header('Prefer' => "return=minimal"); # use return=representation to get full json ←
response
$req->content(JSON::to_json($prefs));
$res = $ua->request($req);
if($res->code != 204) {
    die "Failed to update subscriber preferences: ".$res->decoded_content."\n";
}
say "Updated subscriber preferences";

```

Modifying numbers assigned to a subscriber, changing the password, locking a subscriber, etc. can be done directly on the subscriber resource.

```

$req = HTTP::Request->new('GET', "$uri/api/subscribers/$subscriber_id");
$res = $ua->request($req);
if($res->code != 200) {
    die "Failed to fetch subscriber: ".$res->decoded_content."\n";
}
my $sub = JSON::from_json($res->decoded_content);
delete $sub->{_links}; # not needed in update
push @{$sub->{alias_numbers}}, { cc => 1, ac => 5432, sn => $t }; # add this number
push @{$sub->{alias_numbers}}, { cc => 1, ac => 5433, sn => $t }; # add another number

$req = HTTP::Request->new('PUT', "$uri/api/subscribers/$subscriber_id");
$req->header('Content-Type' => 'application/json');
$req->header('Prefer' => "return=minimal"); # use return=representation to get full json ←
response
$req->content(JSON::to_json($sub));
$res = $ua->request($req);
if($res->code != 204) {
    die "Failed to update subscriber: ".$res->decoded_content."\n";
}
say "Updated subscriber";

```

At the end of a subscriber life cycle, it can be terminated. Once terminated, you can NOT recover the subscriber anymore.

```

$req = HTTP::Request->new('DELETE', "$uri/api/subscribers/$subscriber_id");
$res = $ua->request($req);
if($res->code != 204) {
    die "Failed to terminate subscriber: ".$res->decoded_content."\n";
}
say "Terminated subscriber";

```

Note that certain information is still available in the internal database to perform billing/rating of calls done by this subscriber. Nevertheless, the data is removed from the operational tables of the database, so the subscriber is not able to connect to the system, login or make calls/chats.

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

```

$req = HTTP::Request->new('PATCH', "$uri/api/customers/$customer_id");
$req->header('Content-Type' => 'application/json-patch+json');
$req->header('Prefer' => "return=minimal"); # use return=representation to get full json ↔ response
$req->content(JSON::to_json([
    { op => 'replace', path => '/status', value => 'terminated' }
]));
$res = $ua->request($req); # this will also terminate all still active subscribers
if($res->code != 204) {
    die "Failed to terminate customer: ".$res->decoded_content."\n";
}
say "Terminated customer";

```

## 11.2 API performance considerations

The REST API is designed with pagination support built-in. It is mandatory, to implement pagination in your API clients. If you circumvent pagination by setting the number of rows requested in one API call to a very high number the following side effects may appear:

1. An HTTP timeout at the gateway may occur. The default timeout limit is set to 60s. You can change it by creating a patchtt file for the following template: */etc/ngcp-config/templates/etc/nginx/sites-available/ngcp-panel\_admin\_api.tt2*.
2. Other parts of the system might become unresponsive due to mysql table locks. This especially applies to endpoints related to the Customers entity.

## 12 Configuration Framework

The Sipwise C5 provides a configuration framework for consistent and easy to use low level settings management. A basic usage of the configuration framework only needs two actions already used in previous chapters:

- Edit `/etc/ngcp-config/config.yml` file.
- Execute `ngcpctl apply 'my commit message'` command.

Low level management of the configuration framework might be required by advanced users though. This chapter explains the architecture and usage of Sipwise C5 configuration framework. If the basic usage explained above fits your needs, feel free to skip this chapter and return to it when your requirements change.

A more detailed workflow of the configuration framework for creating a configuration file consists of 7 steps:

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

### 12.1 Configuration templates

The Sipwise C5 provides configuration file templates for most of the services it runs. These templates are stored in the directory `/etc/ngcp-config/templates`.

Example: Template files for `/etc/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.

#### 12.1.1 `.tt2`, `.customtt.tt2` and `.patchtt.tt2` files

These files are the main template files that will be used to generate the final configuration file for the running service. They contain all the configuration options needed for a running Sipwise C5 system. The configuration framework will combine these files with the values provided by `config.yml`, `constants.yml` and `network.yml` to generate the appropriate configuration file.

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

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

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

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

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

Instead of editing *.tt2* files, the configuration framework recognises *.customtt.tt2* files. These files are the same as *.tt2*, but they have higher priority when the configuration framework creates the final configuration files. If you need to introduce changes in a template, you must always copy the required *.tt2* file to *.customtt.tt2*, make changes in the latter file one and leave the *.tt2* file untouched. This way, the system will use the new custom configuration allowing you to switch back to the original one quickly.

Example: We'll create */etc/ngcp-config/templates/etc/lb/kamailio.cfg.customtt.tt2* and use it for our customized configuration. In this example, we'll just append a comment at the end of the template.

```
cd /etc/ngcp-config/templates/etc/kamailio/lb
cp kamailio.cfg.tt2 kamailio.cfg.customtt.tt2
echo '# This is my last line comment' >> kamailio.cfg.customtt.tt2
ngcpconfig apply 'my commit message'
```

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

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



### Warning

users have to upgrade all *.customtt.tt2* manually every time *.tt2* is upgraded, as *ngcpconfig* completely ignores new code in *.tt2* received from new package version.

---

The huge drawback of *.customtt.tt2* files are necessity to keep them up-to-date manually. Keeping them outdated will cause the system misbehaviour as different components will use different code version (as new *.tt2* version will be overwritten by old *.customtt.tt2*).



The `.patchtt.tt2` concept should help users here. It will minimise the manual efforts by using linux "patch" utility. The `ngcpcfg` tool is searching for `.patchtt.tt2` files every time `ngcpcfg build` has been called. If `.patchtt.tt2` is detected, the `ngcpcfg` tool will try to apply `.patchtt.tt2` on `.tt2` and store result in `.customtt.tt2` if no conflicts noticed during patching. Further building process happens in a common way. Example:

```
root@spce:~# ngcpcfg build /etc/kamailio/lb/kamailio.cfg
spce: yml configs were validated successfully
spce: configs were checked successfully
spce: Validating patch '/etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2'
spce: Applying patch '/etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2'
spce: Successfully created '/etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.customtt.tt2'
spce: Requested patchtt operation has finished successfully.
Loading /etc/ngcp-config/config.yml in memory: OK
Loading /etc/ngcp-config/network.yml in memory: OK
Loading /etc/ngcp-config/constants.yml in memory: OK
spce: Generating /etc/kamailio/lb/kamailio.cfg: OK
spce: Executing postbuild for /etc/kamailio/lb/kamailio.cfg
root@spce:~#
```

To convert some/all the current `.customtt.tt2` users can use command `ngcpcfg patch --from-customtt [<customtt_file>]`:

```
root@spce:~# ngcpcfg patch --from-customtt /etc/ngcp-config/templates/etc/kamailio/lb/
kamailio.cfg.customtt.tt2
spce: Validating customtt '/etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.customtt
.tt2'
spce: Creating patchtt file '/etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.
patchtt.tt2'
spce: Requested customtt operation has finished successfully.
root@spce:~#
```

Here is the example of newly created `.patchtt.tt2` file:

```
root@spce:~# cat /etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2
@@ -1799,3 +1799,4 @@
}

# vim: ft=cfg
+# This is my last line comment
root@spce:~#
```

See more details about `.patchtt.tt2` files below in [patchtt](#) section.

---

**Tip**

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

---

### 12.1.2 Using `patchtt` for generation of a relevant `customtt` file

Keeping custom modifications directly in the `.customtt.tt2` templates is NOT recommended as templates become outdated with every software upgrade.

A better way is to handle custom modifications using `.patchtt.tt2` files (e.g. `/etc/ngcp-config/templates/etc/cron.d/cleanup-tools.patchtt.tt2`). In this case, on every "ngcpcfg patch", a `.patchtt.tt2` file will be applied on top of the `.tt2` file and the result will be saved into the `customtt` file and used commonly as described in the previous section. "ngcpcfg patch" is the first step on "ngcpcfg build" that guarantees the latest upstream templates with the availability of the necessary local changes on every configuration apply.

---

**Tip**

The patch to be applied to the corresponding `.tt2` template file is selected in the following order (highest to lowest):  
`*.patchtt.tt2.$HOSTNAME *.patchtt.tt2.$PAIRNAME *.patchtt.tt2.$HA_NODE *.patchtt.tt2`

---

**Note**

If a suitable `patchtt` file is found for a template, then the `ngcpcfg patch` command will overwrite the corresponding `customtt` file, if any.

---

#### 12.1.2.1 Creating a `patchtt` file

Let us see how to introduce custom changes into a template through a `patchtt` file. For example, we need to change the accounting records cleanup time, which is defined in `cleanup-tools.tt2`. Here is how to do this:

- Go to the corresponding templates directory:

```
cd /etc/ngcp-config/templates/etc/cron.d/
```

- Duplicate the required `.tt2` file to `.customtt.tt2`

```
cp ./cleanup-tools.tt2 ./cleanup-tools.customtt.tt2
```

- Introduce the necessary changes to the duplicated file:

```
vim ./cleanup-tools.customtt.tt2
```

- Create the patchtt file from your customtt file and recheck it:

```
ngcpcfg patch --from-customtt ./cleanup-tools.customtt.tt2
cat ./cleanup-tools.patchtt.tt2
```

- Apply and push the changes

```
ngcpcfg apply "Change acc-cleanup time from 00 to 02 hours"
ngcpcfg push all
```

You will notice that the "ngcpcfg apply" command has generated the customtt file for the corresponding template:

```
root@web01a:/etc/ngcp-config/templates/etc/cron.d# ls -l ./cleanup-tools*
-rw----- 1 root root 932 Jan  4 11:11 ./cleanup-tools.customtt.tt2
-rw-r--r-- 1 root root 630 Jan  4 11:08 ./cleanup-tools.patchtt.tt2
-rw-r--r-- 1 root root 932 Dec 18 15:09 ./cleanup-tools.tt2
```

Now, even if cleanup-tools.tt2 slightly changes after a software upgrade, "ngcpcfg apply" will still preserve your custom changes.

---

#### Note

If in a new release the .tt2 file gets changed in the same lines where you had introduced custom changes (e.g. your changes were temporary until a feature is implemented properly in a new software release), the apply process will fail and ask you to review the corresponding .patchtt.tt2 file. Then, check it and either correct it if it is still required or remove it.

---

#### Tip

To convert all existing customtt files to patchtt files use the command: **ngcpcfg patch --from-customtt**

---

### 12.1.3 .prebuild and .postbuild files

After creating the configuration files, the configuration framework can execute some commands before and after placing that file in its target directory. These commands usually are used for changing the file's owner, groups, or any other attributes. There are some rules these commands need to match:

- They have to be placed in a *.prebuild* or *.postbuild* file in the same path as the original *.tt2* file.

- The file name must be the same as the configuration file, but having the mentioned suffixes.
- The commands must be *bash* compatible.
- The commands must return 0 if successful.
- The target configuration file is matched by the environment variable *output\_file*.

Example: We need *www-data* as owner of the configuration file */etc/ngcp-ossbss/provisioning.conf*. The configuration framework will by default create the configuration files with *root:root* as owner:group and with the same permissions (*rwX*) as the original template. For this particular example, we will change the owner of the generated file using the *.postbuild* mechanism.

```
echo 'chgrp www-data ${output_file}' \  
> /etc/ngcp-config/templates/etc/ngcp-ossbss/provisioning.conf.postbuild
```

#### 12.1.4 .services files

*.services* files are pretty similar and might contain commands that will be executed after the *build* process. There are two types of *.services* files:

- The particular one, with the same name as the configuration file it is associated to.  
Example: */etc/ngcp-config/templates/etc/asterisk/sip.conf.services* is associated to */etc/asterisk/sip.conf*
- The general one, named *ngcpcfg.services* that is associated to every file in its target directory.  
Example: */etc/ngcp-config/templates/etc/asterisk/ngcpcfg.services* is associated to every file under */etc/asterisk/*

When the *services* step is triggered all *.services* files associated to a changed configuration file will be executed. In case of the general file, any change to any of the configuration files in the directory will trigger the execution of the commands.

---

#### Tip

If the service script has the execute flags set (*chmod +X \$file*) it will be invoked directly. If it doesn't have execute flags set it will be invoked under *bash*. Make sure the script is *bash* compatible if you do not set execute permissions on the service file.

---

These commands are usually service reload/restarts to ensure the new configuration has been loaded by running services.

---

#### Note

The configuration files mentioned in the following example usually already exist on the platform. Please make sure you don't overwrite any existing files if following this example.

---

Example:

```
echo 'ngcp-service restart mariadb' \  
> /etc/ngcpcfg-config/templates/etc/mysql/my.cnf.services  
echo 'ngcp-service restart asterisk' \  
> /etc/ngcpcfg-config/templates/etc/asterisk/ngcpcfg.services
```

In this example we created two `.services` files. Now, each time we trigger a change to `/etc/mysql/my.cnf` or to `/etc/asterisk/*` we'll see that MySQL or Asterisk services will be restarted by the `ngcpcfg` system.

## 12.2 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.

## 12.3 ngcpcfg and its command line options

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

### 12.3.1 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.

### 12.3.2 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/
```

### 12.3.3 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.

### 12.3.4 decrypt

Decrypt `/etc/ngcp-config-encrypted.tgz.gpg` and restore configuration files, doing the reverse operation of the *encrypt* option. Note: This feature is only available if the `ngcp-ngcpcfg-locker` package is installed.

### 12.3.5 diff

Show uncommitted changes between `ngcpcfg`'s Git repository and the working tree inside `/etc/ngcp-config`. If the tool doesn't report anything it means that there are no uncommitted changes. If the `--addremove` option is specified then new and removed files (iff present) that are not yet (un)registered to the repository will be reported, no further diff actions will be executed then. Note: This option is available since `ngcp-ngcpcfg` version 0.11.0.

### 12.3.6 encrypt

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

### 12.3.7 help

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

### 12.3.8 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.

### 12.3.9 services

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

### 12.3.10 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.

## 13 Network Configuration

Starting with version 2.7, Sipwise C5 uses a dedicated *network.yml* file to configure the IP addresses of the system. The reason for this is to be able to access all IPs of all nodes for all services from any particular node in case of a distributed system on one hand, and in order to be able to generate */etc/network/interfaces* automatically for all nodes based on this central configuration file.

### 13.1 General Structure

The basic structure of the file looks like this:

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

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

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

In Sipwise C5 systems there is only one host entry in the file, and it is always named *self*.

#### 13.1.1 Available Host Options

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



- *role*: The role setting is an array defining which logical roles a node will act as. Possible entries for this setting are:
  - *mgmt*: This entry means the host is acting as management node for the platform. In a Sipwise C5 system this option must always be set. The management node exposes the admin and CSC panels to the users and the APIs to external applications and is used to export CDRs.
  - *lb*: This entry means the host is acting as SIP load-balancer for the platform. In a Sipwise C5 system this option must always be set. The SIP load-balancer acts as an ingress and egress point for all SIP traffic to and from the platform.
  - *proxy*: This entry means the host is acting as SIP proxy for the platform. In a Sipwise C5 system this option must always be set. The SIP proxy acts as registrar, proxy and application server and media relay, and is responsible for providing the features for all subscribers provisioned on it.
  - *db*: This entry means the host is acting as the database node for the platform. In a Sipwise C5 system this option must always be set. The database node exposes the MySQL and Redis databases.
  - *rtplib*: This entry means the host is acting as the RTP relay node for the platform. In a Sipwise C5 system this option must always be set. The RTP relay node runs the *rtplib* Sipwise C5 component.
- *interfaces*: The interfaces setting is an array defining all interface names in the system. The actual interface details are set in the actual interface settings below. It typically includes `lo`, `eth0`, `eth1` physical and a number of virtual interfaces, like: `bond0`, `vlanXXX`
- *<interface name>*: After the interfaces are defined in the *interfaces* setting, each of those interfaces needs to be specified as a separate set of parameters.

Additional main parameters of a node:

- *dbnode*: the sequence number (unique ID) of the node in the database cluster; not used in Sipwise C5 CE systems.
- *status*: one of *online*, *offline*, *inactive*. *inactive* means that the node is up but is not ready to work in the cluster (installing process). *offline* means that the node is not reachable. *online* is a normal working node.

### 13.1.2 Interface Parameters

- *hwaddr*: MAC address of the interface
- *ip*: IPv4 address of the node
- *v6ip*: IPv6 address of the node; optional
- *netmask*: IPv4 netmask
- *v6netmask*: IPv6 netmask
- *gateway*: IPv4 gateway address
- *v6gateway*: IPv6 gateway address
- *advertised\_ip*: the IP address that is used in SIP messages when Sipwise C5 system is behind NAT/SBC. An example of such a deployment is *Amazon AMI*, where the server doesn't have a public IP, so *load-balancer* component of Sipwise C5 needs to know what his public domain is (→ *advertised\_ip*).

- `type`: type of services that the node provides; these are usually the VLANs defined for a particular Sipwise C5 system.

---

#### Note

You can assign a type only once per node.

---

Available types are:

- `api_int`: internal, API-based communication interface. It is used for the internal communication of such services as faxserver, fraud detection and others.
- `aux_ext`: interface for potentially insecure external components like remote system log collection service.
- `rtsp_ext`: main (external) interface for media traffic
- `sip_ext`: main (external) interface for SIP signalling traffic between NGCP and other SIP endpoints
- `sip_ext_incoming`: additional, optional interface for incoming SIP signalling traffic
- `sip_int`: internal SIP interface used by Sipwise C5 components (*lb, proxy, etc.*)
- `ssh_ext`: command line (SSH) remote access interface
- `web_ext`: interface for web-based or API-based provisioning and administration
- `web_int`: interface for the administrator's web panel, his API and generic internal API communication
- `ha_int`: HA (High Availability) communication interface between the services

---

#### Note

Please note that, apart from the standard ones described so far, there might be other *types* defined for a particular Sipwise C5 system.

---

- `vlan_raw_device`: tells which physical interface is used by the particular VLAN
- `post_up`: routes can be defined here (interface-based routing), for example:

```
post_up:
- route add -host 1.2.3.4 gw 192.168.1.1 dev vlan70
- route add -net 10.11.12.0/21 gw 192.168.1.2 dev vlan300
- route del -host 1.2.3.4 gw 192.168.1.1 dev vlan70
- route del -net 10.11.12.0/21 gw 192.168.1.2 dev vlan300
```

- `bond_XY`: specific to "bond0" interface only; these contain Ethernet bonding properties

## 13.2 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.

### 13.2.1 Additional entries in `/etc/hosts`

The file `/etc/hosts` is generated by a template, containing entries for basic host configuration (localhost and basic IPv4/IPv6), and the IPs of other nodes in PRO/CARRIER configurations.

To add extra entries in this file, it can be done in several ways:

- `etc_hosts_global_extra_entries` at the global level, added to all hosts
- `etc_hosts_global_extra_entries` at the host level, which overrides the global one if for some reason the whole content is undesired for a particular host (e.g. to have some but not all of the "default" global entries)
- `etc_hosts_local_extra_entries` at the host level, which are added only to the hosts where this entry is present, if for some reason it is desired to have extra entries only visible in some subset of the hosts

The behaviour is the same in all cases, to append the entries directly to `/etc/hosts`.

Example of both in a configuration file:

```
---
hosts_common:
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
  - 10.100.1.2 server-2 server-2.internal.example.com
hosts:
  db01b:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.db01b
    - 127.0.2.1 local-alias-2.db01b
    - 172.30.52.180 db01b.example.com
    ...
  web01a:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.web01a
    - 127.0.2.1 local-alias-2.web01a
    - 172.30.52.168 web01a.example.com
    etc_hosts_global_extra_entries:
    - 10.100.1.1 server-1 server-1.internal.example.com
    ...
```

With this, the additional output in `/etc/hosts` for `db01b` will be:

```
# local extra entries for host 'db01b'
127.0.1.1 local-alias-1.db01b
127.0.2.1 local-alias-2.db01b
172.30.52.180 db01b.example.com
```

```
# global extra entries
10.100.1.1 server-1 server-1.internal.example.com
10.100.2.1 server-2 server-2.internal.example.com
```

and in web01a:

```
# local extra entries for host 'web01a'
127.0.1.1 local-alias-1.web01a
127.0.2.1 local-alias-2.web01a
172.30.52.168 web01a.example.com

# global extra entries overridden for host 'web01a'
10.100.1.1 server-1 server-1.internal.example.com
```

### 13.2.2 Extra SIP Sockets

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

The correct format consists of a label and value like this:

```
extra_sockets:
  port_5064: udp:10.15.20.108:5064
  test: udp:10.15.20.108:6060
```

The label is shown in the `outbound_socket` peer preference (if you want to route calls to the specific peer out via specific socket); the value must contain a transport specification as in example above (udp, tcp or tls). After adding execute `ngcpcfg apply`:

```
ngcpcfg apply 'added extra socket'
```

The direction of communication through this SIP extra socket is incoming+outgoing. The Sipwise C5 will answer the incoming client registrations and other methods sent to the extra socket. For such incoming communication no configuration is needed. For the outgoing communication the new socket must be selected in the `outbound_socket` peer preference. For more details read the next section Section 13.2.3 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.

---

### 13.2.3 Extra SIP and RTP Sockets

If you want to use an additional interface (with a different IP address) for SIP signalling and RTP traffic you need to add your new interface in the `/etc/network/interfaces` file. Also the interface must be declared in `/etc/ngcp-config/network.yml`.

Suppose we need to add a new SIP socket and a new RTP socket on VLAN 100. You can use the `ngcp-network` tool for adding interfaces without having to manually edit this file:

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

The generated file should look like the following:

```
..
..
  eth0.100:
    hwaddr: ff:ff:ff:ff:ff:ff
    ip: 192.168.1.3
    netmask: 255.255.255.0
    type:
      - sip_ext_incoming
      - rtp_int_100
..
..
  interfaces:
    - lo
    - eth0
    - eth0.100
    - eth1
..
..
```

As you can see from the above example, extra SIP interfaces must have type `sip_ext_incoming`. While `sip_ext` should be listed only once per host, there can be multiple `sip_ext_incoming` interfaces. The direction of communication through this SIP interface is incoming only. The Sipwise C5 will answer the incoming client registrations and other methods sent to this address and remember the interfaces used for clients' registrations to be able to send incoming calls to him from the same interface.

In order to use the interface for the outbound SIP communication it is necessary to add it to `extra_sockets` section in `/etc/ngcp-config/config.yml` and select in the `outbound_socket` peer preference. So if using the above example we want to use the `vlan100` IP as source interface towards a peer, the corresponding section may look like the following:

```
extra_sockets:
  port_5064: udp:10.15.20.108:5064
  test: udp:10.15.20.108:6060
```

```
int_100: udp:192.168.1.3:5060
```

The changes have to be applied:

```
ngcpcfg apply 'added extra SIP and RTP socket'
```

After applying the changes, a new SIP socket will listen on IP `192.168.1.3` and this socket can now be used as source socket to send SIP messages to your peer for example. In above example we used label `int_100`. So the new label "int\_100" is now shown in the `outbound_socket` peer preference.

Also, RTP socket is now listening on `192.168.1.3` and you can choose the new RTP socket to use by setting parameter `rtp_interface` to the Label "int\_100" in your Domain/Subscriber/Peer preferences.

### 13.2.4 Alternative RTP Interface Selection Using ICE

Normally, each interface that was configured with a type that starts with `rtp_` can be selected individually as RTP interface in the Domain/Subscriber/Peer preferences. For example, if the interface types `rtp_ext`, `rtp_int`, and `rtp_int_100` have been configured, the Domain/Subscriber/Peer preferences will allow the RTP interfaces to be selected as either `ext`, `int`, or `int_100` in addition to "default".

The same `rtp_` interface type can be configured on multiple interfaces. If this is the case, and if ICE (*Interactive Connectivity Establishment*) is enabled for a Domain/Subscriber/Peer, it is possible to use ICE to automatically negotiate which interface should be used for RTP communications. ICE must be supported by the remote client for this to work.

For example, `rtp_ext` can be configured on multiple interfaces like so (abbreviated):

```
..
..
  eth0.100:
    type:
      - rtp_ext
..
  eth0.150:
    type:
      - rtp_ext
..
  eth1:
    type:
      - rtp_ext
..
..
```

In this example, the RTP interface `ext` will be available for selection in the Domain/Subscriber/Peer preferences. If selected and if ICE is enabled, the addresses of all three interfaces will be presented to the remote client, and ICE will be used to negotiate

which one of them will be used for communications. This can be useful in multi-homed environments, or when remote clients are on private networks.

### 13.2.5 Extended RTP Port Range Using Multiple Interfaces

If the RTP port range configured via the `config.yml` keys `rtpproxy.minport` and `rtpproxy.maxport` is not sufficient to handle all concurrent calls, it is possible to load-balance the RTP ports across multiple interfaces. This is useful if the RTP proxy runs out of ports and if not enough additional ports are available.

To enable this, multiple interfaces with different addresses must be configured, and interface types of the format `rtp_NAME:SUFFIX` must be assigned to them. For example, if the RTP interface named `ext` should be load-balanced across three interfaces, they can be configured like so (abbreviated):

```
..
..
  eth0.100:
    type:
      - rtp_ext:1
..
  eth0.150:
    type:
      - rtp_ext:2
..
  eth1:
    type:
      - rtp_ext:3
..
..
```

In this example, all three given RTP interface types will be available for selection in the Domain/Subscriber/Peer preferences individually (as `ext:1` and so on), but in addition to that, an interface named just `ext` will also be available for selection. If `ext` is selected, only one of the three RTP interfaces will be selected in a round-robin fashion, thus increasing the number of available RTP ports threefold. The round-robin algorithm only selects an interface if it actually has RTP ports available.

## 14 Licenses

The Sipwise C5—starting from mr5.5.1 release—implements *software licensing* primarily for the commercial products PRO and CARRIER. However as a CE platform operator you may also see a new process running on the system: "licensed". The only purpose of this software module is to collect anonymous statistics about the system usage, namely the following performance indicators are recorded:

- number of provisioned subscribers
- number of registered subscribers
- number of concurrent calls

The **anonymous usage statistics is enabled by default but you can disable it**. In order to do that you have to edit the main configuration file `/etc/ngcp-config/config.yml` and set `general.anonymous_usage_statistics` parameter to `no`. Then apply the new configuration with the usual command: `ngcpcfg apply "Disabled anon. usage stat"`

---

### Tip

If Sipwise C5 operator does not want to have the license client package (`ngcp-license-client` and `ngcp-license-module`) on his system at all, it is possible to replace it with a dummy package: `ngcp-license-client-dummy`. This dummy package does not contain any licensing software, and is available from mr5.5.2 Sipwise C5 release.

---



## 15 Software Upgrade

### 15.1 Release Notes

The Sipwise C5 version mr8.5.3 has the following important changes:

- Add password reset mechanism for Administrators [TT#76108]
- Change the way Lawful Intercept Administrators are managed [TT#76111]
- New Arabic, Hebrew and Dutch voice prompts [TT#80407]
- [PRO/Carrier] New sems routing module [TT#82050]
- [PRO/Carrier] Sems tpcc module High-Availability [TT#77254]
  - [Carrier] Add ability to troubleshoot calls from/to specific subscribers sending SIP messages to an inactive node. [TT#93950]

Some important technical points for those interested:

- Kamailo version upgraded to 5.3.5 [TT#84508]

Please find the complete changelog in our release notes [on our WEB site](#).

### 15.2 Overview

The Sipwise C5 software upgrade procedure to mr8.5.3 will perform several fundamental tasks:

- upgrade the NGCP software packages
- upgrade the NGCP configuration templates
- upgrade the NGCP DB schema
- upgrade the NGCP configuration schema
- upgrade the base system within Debian 10 (buster) to the latest package versions

### 15.3 Pre-upgrade checks

It is recommended to execute the preparatory steps in this chapter a few days before the actual software upgrade. They do not cause a service downtime, so it is safe to execute them during peak hours.

### 15.3.1 Log into the C5 server

Run the terminal multiplexer under the *sipwise* user (to reuse the Sipwise `.screenrc` settings that are convenient for working in multiple windows):

```
screen -S my_screen_name_for_ngcp_upgrade
```

Become root inside your screen session:

```
sudo -s
```

### 15.3.2 Check the overall system status

Check the overall system status:

```
ngcp-status
```

### 15.3.3 Evaluate and update custom modifications

For the below steps, investigate and make sure you understand why the custom modifications were introduced and if they are still required after the software upgrade. If the custom modifications are not required anymore, remove them (e.g. if a bug was fixed in the target release and the existing patch becomes irrelevant).



#### Warning

If you directly change the working configuration (e.g. add custom templates or change the existing ones) for some reason, then the system must be thoroughly tested after these changes have been applied. Continue with the software upgrade preparation only if the results of the tests are acceptable.

---

Find the local changes to the template files:

```
ngcp-customtt-diff-helper
```

The script will also ask you if you would like to download the templates for your target release. To download the new templates separately, execute:

```
ngcp-customtt-diff-helper -d
```

In the tmp folder provided by the script, you can review the patchtt files or merge the current customtt with the new tt2 templates, creating the new customtt.tt2 files. Once you do this, archive the new patchtt/customtt files to reapply your custom modifications after the software upgrade:

```
ngcp-customtt-diff-helper -t
```

Find all available script options with the "-h" parameter.

### 15.3.4 Check system integrity

Check if there are any \*.tt2.dpkg-dist files among the templates. They usually appear when tt2 files are modified directly instead of creating customtt/patchtt files. If you find any \*.tt2.dpkg-dist files, treat the corresponding tt2 files as if they were customtt.tt2 and introduce the changes from the existing tt2 files into the new templates (create associated customtt.tt2 or patchtt.tt2) before the software upgrade.

```
find /etc/ngcp-config -name \*.tt2.dpkg-dist
```

Note that in the end all \*.tt2.dpkg-dist files must be removed before the software upgrade as they prevent the upgrade script from updating the tt2 files.

Check and remove dpkg files left from previous software upgrades.

Make sure that the list is empty before you continue:

```
find /etc/ngcp-config -name \*.tt2.dpkg\*
```

Changes made directly in tt2 templates will be lost after the software upgrade. Only custom changes made in customtt.tt2 or added by patchtt.tt2 files will be kept. Hence, check the system for locally modified tt2 files on **all** nodes:

```
ngcp-status --integrity
```

### 15.3.5 Check the configuration framework status

Check the configuration framework status on **all** nodes. All checks must show the "OK" result and there must be no actions required:

```
ngcpcfg status
```

Run "apt-get update" and ensure that you do not have any warnings and errors in the output.

**Warning**

If the installation uses locally specified mirrors, then the mirrors must be switched to the Sipwise APT repositories (at least for the software upgrade). Otherwise, the public Debian mirrors may not provide packages for old Releases anymore or at least provide outdated ones!

## 15.4 Pre-upgrade steps

**Note**

Sipwise C5 can be upgraded to mr8.5.3 from previous release or previous build only. The script `ngcp-upgrade` will find all the possible destination releases for the upgrade and makes it possible to choose the proper one.

**Note**

If there is an error during the upgrade, the `ngcp-upgrade` script will request you to solve it. Once you've fixed the problem, just 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).

### 15.4.1 `ngcp-upgrade` options

The following options in `ngcp-upgrade` can be specially useful in some instances of upgrade:

- **`--step-by-step`**: confirm before proceeding to next step. With this option the upgrade operation is performed confirming every step before execution, with the possibility to instruct to continue without confirming further steps until the end (if confirmation is only needed for some steps at the beginning).
- **`--pause-before-step STEP_NAME`**: pause execution before step, given by the name of the script (e.g. "backup\_mysql\_db"). This option can be useful in several scenarios, for example:
  - to help to debug problems or work around known problems during upgrades. In this case the operator can pause at a given step known to be problematic or just before a problematic set, perform some manual checks or changes, then continue the upgrade until another step (with confirmation like with the recent option `--step-by-step`), or just continue without stop until the end
  - another use might be to help to speed up upgrades when it involves several nodes: they can all proceed in parallel when it's known to be safe to do so; then perform some parts in lock-step (some nodes waiting until others finish with some stage); then continue in parallel until the end
- **`--skip-db-backup`**: This will speed-up the process in cases where it's deemed unnecessary, and this is very likely in the upgrade of nodes other than the first.

### 15.4.2 Preparing for maintenance mode

Sipwise C5 introduces **Maintenance Mode** with its mr5.4.1 release. The maintenance mode of Sipwise C5 will disable some background services (for instance: *ngcp-mediator*) during the software upgrade. It thus prevents the system from getting into an inconsistent state while the upgrade is being performed. You can activate maintenance mode by applying a simple configuration change as described later.

- Enable maintenance mode:

```
ngcpcfg set /etc/ngcp-config/config.yml "general.maintenance=yes"
```

- Apply configuration changes by executing:

```
ngcpcfg apply 'Enabling maintenance mode before the upgrade to mr8.5.3'
```

### 15.4.3 Set the proper software repositories



#### Warning

Ensure you are using the Sipwise APT repositories.

---

Public Debian mirrors may not provide packages for old Debian releases anymore. Also, they might be outdated. Consider using Sipwise repositories for the time of the upgrade.

## 15.5 Upgrading Sipwise C5 CE

Execute the following commands as *root*:

```
ngcp-prepare-upgrade mr8.5.3
ngcp-upgrade
```

Once up again, double-check your config file */etc/ngcp-config/config.yml* (sections will be rearranged now and will contain more parameters) and your domain/subscriber/peer configuration and test the setup.

## 15.6 Post-upgrade steps

### 15.6.1 Disabling maintenance mode

In order to disable the *maintenance mode*, do the following:

- Disable the maintenance mode:

```
ngcpcfg set /etc/ngcp-config/config.yml "general.maintenance=no"
```

- Apply the changes to configuration templates:

```
ngcpcfg apply 'Disable the maintenance mode after the upgrade to mr8.5.3'
```

### 15.6.2 Post-upgrade checks

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

---

#### Note

You can find a backup of some important configuration files of your existing installation under `/ngcp-data/backup/ngcp-mr8.5.3-*` (where `*` is a place holder for a timestamp) in case you need to roll back something at any time. A log file of the upgrade procedure is available at `/ngcp-data/backup/ngcp-mr8.5.3-*/upgrade.log`.

---

## 15.7 Applying the Latest Hotfixes

If your current release is already the latest or you prefer to be on the LTS release, we still suggest applying the latest hotfixes and critical bug fixes.

Execute all steps as described in Section 15.3. They include the system checks, custommt/patchtt preparation and others. It is important to execute all the steps from the above chapter.

### 15.7.1 Apply hotfixes

```
ngcp-update
```

### 15.7.2 Recheck or update the custom configuration templates

Merge/add the custom configuration templates if needed.

Apply the changes to configuration templates:

```
ngcpconfig apply 'apply customtt/patchtt after installing the latest packages'
```

Execute the final checks as described in the **Post-upgrade checks** section.

## 16 Backup, Recovery and Database Maintenance

### 16.1 Sipwise C5 Backup

For any service provider it is important to maintain a reliable backup policy as it enables prompt services restoration after any force majeure event. Hence, we strongly suggest you to configure a backup procedure. The Sipwise C5 can be integrated with any Debian compatible backup software.

#### 16.1.1 What data to back up

- The database

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

- System configuration

The system configuration folder `/etc/ngcp-config/` must be included in the backup as well. It contains the system specific configuration (like SSL keys). Also you might have some local modifications. We suggest backing up the whole `/etc` folder to preserve the `etckeeper` history to be able to answer when and who changed particular configuration files in the past.

- Exported CDRs (optional)

The `/home/jail/home/cdreexport` 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.

### 16.2 Recovery

In the worst case scenario, when the system needs to be recovered from a total loss, you only need 4 steps to get the services back online:

- Install Sipwise C5 as explained in chapter 2.
- Restore the `/etc/ngcp-config/` directory from the backup, overwriting your local files.
- Restore `mysql.encryption.key` in `constants.yml` if new MariaDB instance/binlogs were encrypted (DB is encrypted by default). See the detailed information in [MariaDB data restoration remarks](#).
- Restore the database from the latest MySQL dump.
- Apply the changes to bring the original configuration into effect:

```
ngcpcfg apply 'restored the system from the backup'
```



## 16.3 Reset Database



### Important

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

---

To reset database to its original state you can use a script provided by CE: \* Execute *ngcp-reset-db*. It will assign new unique passwords for Sipwise C5 services and reset all services. The script will also create dumps for all Sipwise C5 databases.

## 16.4 Accounting Data (CDR) Cleanup

Sipwise Sipwise C5 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 Sipwise C5 components designed for this purpose and they are commonly called *cleantools*. These are basically configurable scripts that interact with NGCP's *accounting* and *kamailio* databases, or remove exported CDR files in order to clean or archive the unnecessary data.

### 16.4.1 Cleanuptools Configuration

The configuration parameters of *cleantools* are located in the main Sipwise C5 configuration file: `/etc/ngcp-config/config.yml`. Please refer to the `config.yml` file description: [Cleanuptools Configuration Data](#) for configuration parameter details.

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

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

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

The Sipwise C5 system administrator can also select the time when cleanup scripts are run, by modifying the schedule here: `/etc/cron.d/cleanup-tools`

### 16.4.2 Accounting Database Cleanup

The script responsible for cleaning up the database is: `ngcp-cleanup-acc`

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

An extract from a sample configuration file is provided here:

```
#####

batch = 10000
archive-target = /ngcp-data/backup/cdr
compress = gzip

username = dbcleaner
password = rcKamRdHhx7saYRbkJfP
host = localhost

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 ngcp-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>`

#### 16.4.2.1 Backup CDRs

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

Configuration values that govern the backup procedure are:

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



#### Important

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

---

- `backup-retro`: How many months to process for backups, going backwards in time. Using the example above, with this value set to "3", the months October, September and August would be backed up, while any older records would be left untouched.

#### 16.4.2.2 Archive CDRs

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

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

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



#### Important

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

---

- `archive-target`: Target directory for writing the SQL dump files into. If explicitly specified as `"/dev/null"`, then no actual archiving will be performed, but instead the tables will only be dropped from database.

- `compress`: If set to "gzip", then gzip the dump files after creation. If unset, do not compress.
- `host`, `username` and `password`: As dumping is performed by an external command, those variables are reused from the `connect` statement.

### 16.4.2.3 Cleanup CDRs

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

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

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

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

### 16.4.3 Exported CDR Cleanup

The script responsible for cleaning up exported CDR files is: `ngcp-cleanup-cdr-files`

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

A sample configuration file is provided here:

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

```
path: /home/jail/home/cdreexport/system/20[0-9][0-9][0-9][0-9]/[0-9][0-9]
wildcard: yes
remove_empty_directories: yes
max_age_days: ~
```

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

Configuration values that define the files to be deleted:

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

## 17 Platform Security, Performance and Troubleshooting

Once Sipwise C5 is in production, security and maintenance becomes really important. In this chapter, we'll go through a set of best practices for any production system.

### 17.1 Sipwise SSH access to Sipwise C5

The Sipwise C5 provides SSH access to the system for Sipwise operational team for debugging and final tuning. Operational team uses user *sipwise* which can be logged in through SSH key only (password access is disabled) from dedicated access server *jump.sipwise.com* only.

To completely remove Sipwise access to your system, please execute as user root:

```
root@myserver:~# ngcp-support-access --disable && apt-get install ngcp-support-noaccess
```

---

#### Note

you have to execute the command above on each node of your Sipwise C5 system!

---



#### Warning

please ensure that the script complete successfully:

---

```
* Support access successfully disabled.
```

If you need to restore Sipwise access to the system, please execute as user root:

```
root@myserver:~# apt-get install ngcp-support-access && ngcp-support-access --enable
```



#### Warning

please ensure that the script complete successfully:

---

```
* Support access successfully enabled.
```

## 17.2 Firewalling

### 17.2.1 Firewall framework

The Sipwise C5 runs a wide range of services. In order to secure the platform while allowing access to Sipwise C5, Sipwise C5 configuration framework provides a set of predefined network zones. Services are aggregated into appropriate zones by default. Zones are assigned to network interfaces (and VLANs if applicable) in `/etc/ngcp-config/network.yml`.

#### Caution



Though the default firewall setup provided by Sipwise C5 configuration framework provides a safe setup for Sipwise C5, security audits of the platform performed by qualified engineers before commissioning the platform into service are strongly recommended. Customization of the setup requires in-depth knowledge of firewalling principles in general and the *netfilter* facility in particular.

Table 16: Sipwise C5 network zones

| Zone name | Description   |
|-----------|---|
| ha_int    | Internal HA (High Availability) communication interface between the services                        |
| rtp_ext   | Interface for external RTP media relay between Sipwise C5 and endpoints (e.g. user agents, peers)   |
| sip_ext   | Interface for external SIP signalling between Sipwise C5 and endpoints (e.g. user agents, peers)    |
| sip_int   | Interface for internal signalling, e.g. between load-balancers, proxies and applications servers    |
| ssh_ext   | Interface providing external access to Sipwise C5 command line interface                            |
| web_ext   | Interface providing access to the customers' self-care Web panel                                    |
| web_int   | Interface for access to the administrative Web panel, its REST APIs and internal API communications |

#### Note

Additional custom zones may be configured, but will not be automatically integrated into the firewall configuration.

To facilitate firewall functionality, Sipwise C5 uses the Kernel's *netfilter* facility and *iptables-persistent* as an interface to *netfilter*. *Netfilter* is using *tables* and within that *chains* to store rules in this hierarchy: *table* → *chain* → *rule*. Default firewall setups of Sipwise C5 do not use netfilter tables *nat* and *raw*, but only default table *filter*.

#### Note

Custom *nat* rules for IPv4 and IPv6 may be added in file `/etc/ngcp-config/config.yml` in sections `security→firewall→nat_rules4` and `security→firewall→nat_rules6`.

Each *chain* deploys a *default policy* handling packets which did not trigger and rule in a particular *chain*.

Table 17: Sipwise C5 *netfilter* default policies

| Chain     | Default policy | Description  |
|-----------|----------------|--|
| INPUT     | DROP           | Handling all packets directly destined for a Sipwise C5 node (only packets matching a rule are allowed)                        |
| FORWARD   | DROP           | Handling all packets received by a Sipwise C5 node and destined for another, non-local IP destination (no default rules added) |
| OUTPUT    | ACCEPT         | Handling all packets originating on a Sipwise C5 node (no default rules added)   |
| rtpengine | N/A            | Container for rtpengine rule to allow the rule to persist even when the Kernel module is unloaded (e.g. during upgrades)       |

The default firewall setup provided by Sipwise C5:

- adds rules to INPUT to secure access to platform and services
- blocks all traffic from and to FORWARD
- allows all OUTPUT traffic

### 17.2.2 Sipwise C5 firewall configuration

The Sipwise C5 comes with a preconfigured set of firewall rules, which can be enabled and configured in `/etc/ngcp-config/config.yml` in section `security→firewall`. Refer to Section [B.1.29](#) 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 Sipwise C5 firewall subsystem by default is disabled in `/etc/ngcp-config/config.yml` key `security.firewall.enable: no`. This is to avoid blocking any traffic inadvertently during installation. After the firewall subsystem has been configured appropriately, it needs to be enabled by setting `security.firewall.enable: yes` in `/etc/ngcp-config/config.yml`.

---



### 17.2.3 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 18: Firewall system rules

| Zone    | Chain        | Target    | Rule   | Description   |
|---------|--------------|-----------|--|---|
| all     | INPUT        | rtpengine | <code>-p udp -j rtpengine</code>                                   | 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 | <code>-p udp -j RTPENGINE --id 0</code>                            | Feeds all RTP packets to RTPENGINE Kernel module  |
| n/a     | INPUT        | ACCEPT    | <code>-i lo -j ACCEPT</code>                                       | Accept all packets received by local loopback interface   |
| all     | INPUT        | ACCEPT    | <code>-m state --state RELATED, ESTABLISHED -j ACCEPT</code>       | Accept all incoming packets tied to <i>related</i> or <i>established</i> connections  |
| all     | INPUT (IPv4) | ACCEPT    | <code>-p icmp -m icmp --icmp-type 8 -j ACCEPT</code>               | Accept all ICMP <i>echo</i> messages  |
| all     | INPUT (IPv4) | ACCEPT    | <code>-p icmp -m icmp --icmp-type 0 -j ACCEPT</code>               | Accept all ICMP <i>echo reply</i> messages  |
| all     | INPUT (IPv6) | ACCEPT    | <code>-A INPUT -p ipv6-icmp -j ACCEPT</code>                       | Accept all ICMPv6 messages  |
| all     | INPUT        | cluster   | <code>-j cluster</code>  | Divert all incoming packets to the <i>cluster</i> chain   |
| all     | cluster      | ACCEPT    | <code>-s &lt;node_ip&gt; -j ACCEPT</code>                          | Set of rules white-listing all IP-addresses owned by Sipwise C5 platform for incoming traffic   |
| api_int | INPUT        | ACCEPT    | <code>-p tcp --dport &lt;ossbss.port&gt; -j ACCEPT</code>          | Set of rules for all <i>api_int</i> interfaces accepting all incoming packets for API port defined in <code>/etc/ngcp-config/config.yml</code> with key <i>ossbss.port</i>                                  |
| mon_ext | INPUT        | ACCEPT    | <code>+p udp -s &lt;snmpclient_ip&gt; --dport 161 -j ACCEPT</code> | Set of rules for all <i>mon_ext</i> interfaces based on a list of IPs for all SNMP communities configured in <i>snmpd.communities</i>   |

Table 18: (continued)

| Zone    | Chain | Target              | Rule   | Description  |
|---------|-------|---------------------|--|--|
| rtp_ext | INPUT | ACCEPT/ <i>name</i> | <code>-p udp --dport<br/>&lt;<i>rtpproxy.minport</i>&gt;:'&lt;rtpproxy.minport&gt;'<br/>-j ACCEPT/<i>name</i></code> | Set of rules for all <i>rtp_ext</i> interfaces accepting all incoming packets for RTP port range defined in <i>/etc/ngcp-config/config.yml</i> with keys <i>rtpproxy.minport</i> and <i>rtpproxy.maxport</i> (see note below for custom options) |
| sip_ext | INPUT | ACCEPT              | <code>-p udp --dport<br/>&lt;<i>kamailio.lb.port</i>&gt; -j<br/>ACCEPT</code>  | Set of rules for all <i>sip_ext</i> interfaces accepting all packets on the loda balancer's SIP signalling port defined in <i>/etc/ngcp-config/config.yml</i> with key <i>kamailio.lb.port</i> (UDP)   |
| sip_ext | INPUT | ACCEPT              | <code>-p tcp --dport<br/>&lt;<i>kamailio.lb.port</i>&gt; -j<br/>ACCEPT</code>  | Set of rules for all <i>sip_ext</i> interfaces accepting all packets on the loda balancer's SIP signalling port defined in <i>/etc/ngcp-config/config.yml</i> with key <i>kamailio.lb.port</i> (TCP)   |
| sip_ext | INPUT | ACCEPT              | <code>-p tcp --dport<br/>&lt;<i>kamailio.lb.tls.port</i>&gt;<br/>-j ACCEPT</code>                                    | Set of rules for all <i>sip_ext</i> interfaces accepting all packets on the loda balancer's SIP signalling port defined in <i>/etc/ngcp-config/config.yml</i> with key <i>kamailio.lb.tls.port</i> (TCP/TLS)                                     |
| sip_ext | INPUT | ACCEPT              | <code>-p tcp --dport 5222 -j<br/>ACCEPT</code>   | Set of rules for all <i>sip_ext</i> interfaces accepting all packets on TCP port 5222 (XMPP client)  |
| sip_ext | INPUT | ACCEPT              | <code>-p tcp --dport 5269 -j<br/>ACCEPT</code>   | Set of rules for all <i>sip_ext</i> interfaces accepting all packets on TCP port 5269 (XMPP server)  |
| sip_ext | INPUT | ACCEPT              | <code>-p tcp --dport<br/>&lt;<i>pushd.port</i>&gt; -j ACCEPT</code>  | Set of rules for all <i>sip_ext</i> interfaces accepting all packets incoming for the <i>pushd</i> server port configured in <i>/etc/ngcp-config/config.yml</i> with key <i>pushd.port</i>   |

Table 18: (continued)

| Zone    | Chain | Target | Rule  | Description   |
|---------|-------|--------|---|---|
| ssh_ext | INPUT | ACCEPT | <pre>-A INPUT -i &lt;ssh_ext_interface&gt; -p tcp -s &lt;sshd.permit_support_from&gt; --dport sshd.port -j ACCEPT</pre> | List of rules to accept incoming packets for SSH on all <i>ssh_ext</i> interfaces from hosts configured in <i>/etc/ngcp-config/config.yml</i> with key <i>sshd.permit_support_from</i>                      |
| web_ext | INPUT | ACCEPT | <pre>-p tcp --dport &lt;www_admin.http_csc.port&gt; -j ACCEPT</pre>   | List of rules to accept incoming packets for the <i>Customer Self Care</i> interface defined in <i>/etc/ngcp-config/config.yml</i> with key <i>www_admin.http_csc.port</i> on all <i>web_ext</i> interfaces |
| web_int | INPUT | ACCEPT | <pre>-p tcp --dport &lt;www_admin.http_admin.port&gt; -j ACCEPT</pre>   | List of rules to accept incoming packets for the <i>Admin Panel</i> interface defined in <i>/etc/ngcp-config/config.yml</i> with key <i>www_admin.http_admin.port</i> on all <i>web_int</i> interfaces      |

**Caution**

To function correctly, the *rtengine* requires an additional *iptables* rule installed. This rule (with a target of *RTPENGINE*) is automatically installed and removed when the *rtengine* starts and stops, so normally you don't need to worry about it. However, any 3rd party firewall solution can potentially flush out all existing *iptables* rules before installing its own, which would leave the system without the required *RTPENGINE* rule and this would lead to decreased performance. It is imperative that any 3rd party firewall solution either leaves this rule untouched, or installs it back into place after flushing all rules out. The complete parameters to install this rule (which needs to go into the *INPUT* chain of the *filter* table) are: `-p udp -j RTPENGINE --id 0`

**Note**

Some of the parameters used to populate the firewall rules automatically may contain hostnames instead of IP addresses. Since firewall rules need to be configured based on IP addresses by design, Sipwise C5 configuration framework will lookup such hostnames during *ngcpconfig apply* and expand them to the IP addresses as returned by *gethostbyname*. If DNS resolving changes for such hostnames due to changes to DNS the rules will not update automatically. Another run of *ngcpconfig apply* will be needed to reperform the lookup and update the rules to reflect changes in DNS. If this step is omitted, clients may be locked out of the system.

**Note**

By default, the rules for the `rtp_ext` zone are created with a target of `ACCEPT`. It is optionally possible to create these rules with another `iptables` chain as target, and instruct the RTP proxy to dynamically manage individual rules for each running call in this chain. If this is enabled, the chain with the name given in the `/etc/ngcp-config/config.yml` key `rtpproxy→firewall_iptables_chain` will be created as empty, leaving the effective target for UDP packets within the RTP port range as the table's default policy (normally `DROP`). The RTP proxy will then dynamically create one `ACCEPT` rule for each open RTP media port in the given chain when a call starts, and delete it when the call is finished. It should be noted that dynamically creating and deleting `iptables` rules can incur a significant performance overhead, especially in scenarios with high call volumes, and it is therefore not recommended to enable this feature in such cases.

**17.2.4 Custom rules**

The Sipwise C5 configuration framework makes it possible to add custom rules to the firewall setup in `/etc/ngcp-config/config.yml`. The custom rules are added after the system rules. Hence, they apply for packets not matched by the system 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.

## 17.2.5 Example firewall configuration section

An example for Sipwise C5 firewall configuration in `/etc/ngcp-config/config.yml` enabling both the firewall subsystem and the logging facility may look like:

```
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'
```

## 17.3 Password management

The Sipwise C5 comes with some default passwords the user should change during the deployment of the system. They have been explained in the previous chapters of this handbook.



### Important

Many Sipwise C5 services use MySQL backend. Users and passwords for these services are created during the installation. These passwords are unique for each installation, and the connections are restricted to localhost. You should not change these users and passwords.

---

### 17.3.1 The "root" account

The Sipwise C5's super-user account comes with a preconfigured password. It is imperative that this password is changed by the operator immediately after Sipwise C5 is shipped and before it is connected to any potentially unsecure public or private network using a secure password in compliance with existing password policies of the operator. The "root" password must not be shared outside of the operator's organization including Sipwise engineers. The "root" password must not be shared in any publicly accessible communications including e-mail or ticketing systems.

To change the root password log into the freshly deployed system as "root" using the preconfigured password and execute:

```
root@myserver:~# passwd
```

Then follow the prompts to change the password.

The Vagrant/VirtualBox/VMWare Sipwise C5 images come with more default credentials which should be changed immediately:

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

### 17.3.2 The "administrator" account

The Sipwise C5 Web-interface comes with a preconfigured "administrator" account deployed with a default password. This account can be considered Sipwise C5 application super-user and has far-reaching access to application specific settings via the Web-interface. It is imperative that the password for this account is changed by the operator immediately after Sipwise C5 is shipped and before it is connected to any potentially unsecure public or private network using a secure password in compliance with existing password policies of the operator. The "administrator" password must not be shared outside of the operator's organization including Sipwise engineers. The "administrator" password must not be shared in any publicly accessible communications including e-mail or ticketing systems.

The password for the "administrator" account can be changed via the Web-interface.

### 17.3.3 The "cdreexport" account

The login for the system account *cdreexport* is disabled by default. Although this is a jailed account, it has access to sensitive information, namely the Call Detail Records of all calls. SSH keys should be used to login this user, or alternatively a really strong password should be used when setting the password via *passwd cdreexport*.

### 17.3.4 The MySQL "root" user

The *root* user in MySQL has no default password. A password should be set using the *mysqladmin password* command.

### 17.3.5 The "ngcpssoap" account

Generate new password for user *ngcpssoap* to access the provisioning interfaces, see the details in Section 11.

## 17.4 Remote *root* logins via SSH

To mitigate possible system intrusions from the outside, it is commonly held best practise to disable remote *root* (administrator) logins. With remote root logins disabled, it's still possible to log into the system via SSH to a regular, non-privileged user account, and then use *sudo* or *su* to elevate account privileges to *root* administrator level.

**Warning**

For system setup purposes, the default setting on the Sipwise C5 is to permit remote *root* logins via SSH. It is strongly recommended to change this default setting as soon as possible.

Once you've created a user account for yourself that can be used to gain *root* privileges, remote *root* logins can be disabled via the config switch `sshd→permit_root_login` in `/etc/ngcp-config/config.yml`.

```
sshd:  
  permit_root_login: yes
```

Valid options are:

- *yes* - permit remote root logins via SSH. Not recommended.
- *no* - prohibit all remote root logins via SSH.
- *prohibit-password* - permit remote root logins, except when password authentication is used. This is a good setting if you still wish to access the *root* account directly using public-key authentication, but also want to prohibit remote brute-force attacks against the *root* account password.
- *forced-commands-only* - identical to *prohibit-password* but with the additional restriction that only SSH config sections that have a forced command (via `command=` or `ForceCommand`) are permitted to log in. For advanced users.

## 17.5 *sudo-io*: logging input/output of commands run through *sudo*

It is possible to enable logging for the input and output of commands run through *sudo*, along other meta-information like timing, to be able to reproduce sessions.

The logs are saved in directories with a special structure, the numbers are for a so-called "session" or sequence:

```
$BASE_DIR/$USERNAME/00/00/00
```

For example:

```
/var/log/sudo-io/root/00/00/00
```

and within them, several files for tty input and output, stdin/stdout/stderr, and other ancillary files.

New session use `00/00/01`, `00/00/02` and so on.

This is disabled by default, but can be configured in `/etc/ngcp-config/config.yml` with the following options:

```
sudo:
  logging:
    enable: no
    exclude_users: []
    max_sessions: 0
```

Valid options are:

- *enable*: *no* (default) or *yes*; to use the feature or not
- *exclude\_users*: list of users whose commands are excluded from logging
- *max\_sessions*: 0 by default, which means unlimited. If set to a positive integer, the sequence returns to 0 after reaching it, and starts to overwrite the files in the subdir *00/00/00* instead of increasing indefinitely.

**Warning**

This sudo plugin saves all input and output including passwords typed in shell prompts (even when they are not echoed to the screen), and this is unencrypted, so please consider the security implications.

---

**Warning**

Even if in some versions the compression of these files is enabled (releases mr9.5 and after), some commands can use huge amounts of data and fill-up the available disk space very quickly, and full disk will cause serious problems (failures in services, or even preventing external access to the system to fix the problems). Please consider setting up monitoring of free space, or mitigating measures like setting a small amount of sessions saved, taking data out the system often for preservation and deleting sessions from previous days, etc.

---

**Warning**

Additionally, this has a performance impact (both CPU and I/O) on every command run with sudo that triggers logging, negligible for most commands but significant when the input or output is large.

---

## 17.6 SSL certificates

The Sipwise C5 provides default, self-signed SSL certificates for SSL connections. These certificates are common for every installation. Before going to production state, the system administrator should provide SSL certificates for the web services. These certificates can either be shared by all web interfaces (*provisioning*, *administrator interface* and *customer self care interface*), or separate ones for each them can be used.

- Generate the certificates. The *customer self care interface* certificate should be signed by a certification authority to avoid browser warnings.
- Upload the certificates to the system



- Set the path to the new certificates in `/etc/ngcp-config/config.yml`:
  - `ossbss→apache→autoprov→sslcertfile` and `ossbss→apache→autoprov→sslcertkeyfile` for the *provisioning interface*.
  - `ossbss→apache→restapi→sslcertfile` and `ossbss→apache→restapi→sslcertkeyfile` for the *REST interface*.
  - `www_admin→http_admin→sslcertfile` and `www_admin→http_admin→sslcertkeyfile` for the *admin interface*.
  - `www_admin→http_csc→sslcertfile` and `www_admin→http_csc→sslcertkeyfile` for the *customer self care interface*.
- Apply the configuration changes with `ngcpcfg apply 'added web ssl certs'`.

The Sipwise C5 also provides the self-signed SSL certificates for SIP over TLS services. The system administrator should replace them with certificates signed by a trusted certificate authority if he is going to enable it for the production usage (`kamailio→lb→tls→enable` (disabled by default)).

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

## 17.7 Securing your Sipwise C5 against SIP attacks

The Sipwise C5 allows you to protect your VoIP system against SIP attacks, in particular **Denial of Service** and **brute-force attacks**. Let's go through each of those attacks and let's see how to configure your system in order to face such situations and react against them.

### 17.7.1 Denial of Service

As soon as you have packets arriving on your Sipwise C5 server, it will require a bit of time of your CPU. Denial of Service attacks are aimed to break down your system by sending floods of SIP messages in a very short period of time and keep your system busy to handle such huge amount of requests. Sipwise C5 allows you to block such kind of attacks quite easily, by configuring the following section in your `/etc/ngcp-config/config.yml` :

```
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 Sipwise C5 receives more than 50 messages from the same IP in a time window of 2 seconds, that IP will be blocked for 3600 sec, and you will see in the `kamailio-lb.log` a line saying:

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

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

```
# ngcp-kamctl lb fifo htable.dump ipban
```

### Excluding SIP endpoints from banning

There may be some SIP endpoints that send a huge traffic towards Sipwise C5 from a specific IP address. A typical example is a *SIP Peering Server*.



#### Caution

Sipwise C5 supports handling such situations by excluding all defined *SIP Peering Servers* from DoS protection mechanism.

---

The Sipwise C5 platform administrator may also add whitelisted IP addresses manually in `/etc/ngcp-config/config.yml` at `kamailio.lb.security.dos_whitelisted_ips` and `kamailio.lb.security.dos_whitelisted_subnets` parameters.

### 17.7.2 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 Sipwise C5 allows you to detect and block such attacks quite easily, by configuring the following `/etc/ngcp-config/config.yml` section:

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

You may increase the number of failed attempt if you want (in some cases it's better to be safed, some users can be banned accidentally because they are not writing the right password) and adjust the ban time. If a user try to authenticate an INVITE/REGISTER (or more in general any request containing an "Authorization" or "Proxy-Authorization" SIP header) and it fails more then 3 times, the "user@domain" (not the IP as for Denial of Service attack) will be blocked for 3600 seconds (see `failed_auth_ban_time` on `/etc/ngcp-config/config.yml`). In this case you will see 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 htable.dump auth
```

## 17.8 Topology Hiding

### 17.8.1 Introduction to Topology Hiding on NGCP

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

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

### 17.8.2 Topology Masking Mechanism

Concealment of sensitive information using this mechanism is achieved through encoding the original content of selected SIP headers. Then Sipwise C5 will create a new SIP URI using a preselected IP address and the encoded content as URI parameter, finally re-assembling the SIP header.

Examples for encoded SIP headers:

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

The *load-balancer* element of Sipwise C5 has an SBC role, from the SIP peers point of view. The *LB* offers topology masking function that can be simply activated through a configuration change. By default the function is disabled.

#### 17.8.2.1 Configuration of Topology Masking

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

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

Meaning of the configuration parameters:

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

---

**Tip**

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

---

### 17.8.2.2 Considerations for Topology Masking

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

The most common example is the consequence that **SIP message size may grow** when applying topology masking. The fact that SIP messages become larger may even prevent Sipwise C5 from communicating successfully with another SIP entity (a peer SBC, for example). This can be expected under following circumstances:

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

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

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

### 17.8.3 Topology Hiding Mechanism

This mechanism achieves topology hiding by stripping the SIP routing headers that show topology details and storing those data in the associative data structure (hash) in the Redis DB so that it can look it up when a reply or in-dialog SIP message comes in. From the signaling perspective it simulates a SBC (Session Border Controller) on the LB.

### 17.8.3.1 Considerations for Topology Hiding

This mechanism offers some benefits over the older topology masking approach:

- It enables the Sipwise C5 to interconnect with SIP endpoints that are not capable of operating through a SIP proxy.
- The message size is decreased because of stripping the SIP Record-Route, Route and Via header fields.
- It solves the interoperability issues with SIP ALG in some cases.
- It retains also the lightweight nature and the efficient operation.

The module uses the auto-expiration of the Redis keys so it can cause temporary spikes in the memory usage and redis keys count until produced data is cleaned up by redis.

### 17.8.3.2 Configuration of Topology Hiding

Activation of the topology hiding function is done through the modification of the following configuration parameters in `/etc/ngcp-conf` file (shown below with default values of parameters):

```
topos:
  enable: no
  redis_db: 24
```

In order to activate the function, you should set `enable: 'yes'` in `/etc/ngcp-config/config.yml` and leave the Redis DB number unchanged, then execute `ngcpcfg apply "activated topos"`.

## 17.9 System Requirements and Performance

The Sipwise C5 is a very flexible system, capable of serving from hundreds to several tens of thousands of subscribers in a single node. The system comes with a default configuration, capable of serving up to 50.000 subscribers in a *normal* environment. But there is no such thing as a *normal* environment. And Sipwise C5 has sometimes to be tuned for special environments, special hardware requirements or 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 Sipwise C5 administrator tune the system requirements for optimum performance.

Table 19: Requirement\_options

| Option                          | Default value           | Requirement impact   |
|---------------------------------|-------------------------|--|
| cleanuptools→binlog_days        | 15                      | Heavy impact on the harddisk storage needed for mysql logs. It can help to restore the database from backups or restore broken replication.  |
| database→bufferpoolsize         | 64MB                    | For test systems or low RAM systems, lowering this setting is one of the most effective ways of releasing RAM. The administrator can check the innodb buffer hit rate on production systems; a hit rate over 99% is desired to avoid bottlenecks.  |
| kamailio→lb→pkg_mem             | 16                      | This setting affects the amount of RAM the system will use. Each kamailio-lb worker will have this amount of RAM reserved. Lowering this setting up to 8 will help to release some memory depending on the number of kamailio-lb workers running. This can be a dangerous setting as the lb process could run out of memory. Use with caution.   |
| kamailio→lb→shm_mem             | 1/16 * Total System RAM | The installer will set this value to 1/16 of the total system RAM. This setting does not change even if the system RAM does so it's up to the administrator to tune it. It has been calculated that 1024 (1GB) is a good value for 50K subscriber environment. For a test environment, setting the value to 64 should be enough. "Out of memory" messages in the kamailio log can indicate that this value needs to be raised. |
| kamailio→lb→tcp_children        | 8                       | Number of TCP workers kamailio-lb will spawn per listening socket. The value should be fine for a mixed UDP-TCP 50K subscriber system. Lowering this setting can free some RAM as the number of kamailio processes would decrease. For a test system or a pure UDP subscriber system 2 is a good value. 1 or 2 TCP workers are always needed.  |
| kamailio→lb→tls→enable          | yes                     | Enable or not TLS signaling on the system. Setting this value to "no" will prevent kamailio to spawn TLS listening workers and free some RAM.  |
| kamailio→lb→udp_children        | 8                       | See <i>kamailio→lb→tcp_children</i> explanation  |
| kamailio→proxy→children         | 8                       | See <i>kamailio→lb→tcp_children</i> 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_interval | 30                      | Interval for the proxy to send a NAT keepalive OPTIONS message to the nated subscriber. If decreased, this setting will increase the number of OPTIONS requests the proxy needs to send and can impact in the network traffic and the number of natping processes the system needs to run. See <i>kamailio→proxy→natping_processes</i> explanation.  |

Table 19: (continued)

| Option  | Default value           | Requirement impact   |
|---|-------------------------|--|
| <code>kamailio→proxy→natping_processes</code> | 7                       | Kamailio-proxy will spawn this number of processes to send keepalive OPTIONS to the nated subscribers. Each worker can handle about 250 messages/second (depends on the hardware). Depending the number of nated subscribers and the <code>kamailio→proxy→natping_interval</code> parameter the number of workers may need to be adjusted. The number can be calculated like $\text{nated\_subscribers}/\text{natping\_interval}/\text{pings\_per\_second\_per\_process}$ . For the default options, assuming 50K nated subscribers in the system the parameter value would be $50.000/30/250 = (6,66)$ 7 workers. 7 is the maximum number of processes kamailio will accept. Raising this value will cause kamailio not to start. |
| <code>kamailio→proxy→shm_mem</code>           | 1/16 * Total System RAM | See <code>kamailio→lb→shm_mem</code> explanation.  |
| <code>rateomat→enable</code>                  | yes                     | Set this to no if the system shouldn't perform rating on the CDRs. This will save CPU usage.   |
| <code>rsyslog→external_log</code>             | 0                       | If enabled, the system will send the log messages to an external server. Depending on the <code>rsyslog→external_loglevel</code> parameter this can increase dramatically the network traffic.   |
| <code>rsyslog→ngcp_logs_preserve_days</code>  | 93                      | This setting will set the number of days ngcp logs under <code>/var/log/ngcp</code> will be kept in disk. Lowering this setting will free a high amount of disk space.   |

**Tip**

In case of using virtualized environment with limited amount of hardware resources, you can use the script `ngcp-toggle-performance-config` to adjust Sipwise C5 configuration for high/low performance:

```
root@spce:~# /usr/sbin/ngcp-toggle-performance-config
/usr/sbin/ngcp-toggle-performance-config - tool to adjust Sipwise C5 configuration for low ↔
/high performance

--help           Display this usage information
--high-performance Adjust configuration for system with normal/high performance
--low-performance Adjust configuration for system with low performance (e.g. VMs)

root@spce:~#
```

## 17.10 Troubleshooting

The Sipwise C5 platform provides detailed logging and log files for each component included in the system via rsyslog. The main folder for log files is `/var/log/ngcp/`, it contains a list of self explanatory log files named by component name.

The Sipwise C5 is a high performance system which requires compromise between traceability (maximum amount of debug information being written to hard drive) and productivity (minimum load on IO subsystem). This is the reason why different log levels are configured for the provided components by default.

Most log files are designed for debugging Sipwise C5 by Sipwise operational team while main log files for daily routine usage are:

| Log file   | Content   | Estimated size |
|--|---|----------------|
| <code>/var/log/ngcp/api.log</code>   | API logs providing type and content of API requests and responses as well as potential errors   | medium         |
| <code>/var/log/ngcp/panel.log</code><br><code>/var/log/ngcp/panel-debug.log</code> | Admin Web UI logs when performing operational tasks on the ngcp-panel   | medium         |
| <code>/var/log/ngcp/cdr.log</code>   | mediation and rating logs, e.g. how many CDRs have been generated and potential errors in case of CDR generation or rating fails for particular accounting data | medium         |



| Log file                         | Content   | Estimated size |
|----------------------------------|---|----------------|
| /var/log/ngcp/kamailio-proxy.log | Overview of SIP requests and replies between lb, proxy and sems processes. It's the main log file for SIP overview      | huge           |
| /var/log/ngcp/kamailio-lb.log    | Overview of SIP requests and replies along with network source and destination information flowing through the platform | huge           |
| /var/log/ngcp/sems.log           | Overview of SIP requests and replies between lb, proxy and sems processes   | small          |
| /var/log/ngcp/rtp.log            | rtpengine related log, showing information about RTP communication  | small          |

**Warning**

it is highly NOT recommended to change default log levels as it can cause system IO overloading which will affect call processing.

**Note**

the exact size of log files depend on system type, system load, system health status and system configuration, so cannot be estimated with high precision. Additionally operational network parameters like ASR and ALOC may impact the log files' size significantly.

**17.10.1 Collecting call information from logs**

The easiest way to fetch information about a single call among the log files is the search for the SIP CallID (a unique identifier for a SIP dialog). The call ID is used as call marker in almost all the VoIP related log file, such as `/var/log/ngcp/kamailio-lb.log`, `/var/log/ngcp/kamailio-proxy.log`, `/var/log/ngcp/sems.log` or `/var/log/ngcp/rtp.log`. Example of kamailio-proxy.log line:

```
Nov 19 00:35:56 sp1 proxy[7475]: NOTICE: <script>: New request on proxy - M=REGISTER R=sip: ←
sipwise.local
F=sip:jdoe@sipwise.local T=sip:jdoe@sipwise.local IP=10.10.1.10:5060 (127.0.0.1:5060) ID ←
=364e4676776621034977934e055d19ea@127.0.0.1 UA='SIP-UA 1.2.3.4'
```

The above line shows the SIP information you can find in a general line contained in `/var/log/ngcp/kamailio-*`:

- M=REGISTER : The SIP Method
- R=sip:sipwise.local : The SIP Request URI
- F=sip:jdoe@sipwise.local : The SIP From header
- T=sip:jdoe@sipwise.local : The SIP To header
- IP=10.10.1.10:5060 (127.0.0.1:5060) : The source IP where the message is coming from. Between brackets it is shown the local internal IP where the message come from (in this case Load Balancer)
- ID=364e4676776621034977934e055d19ea@127.0.0.1 : The SIP CallID.
- UAIP=10.10.1.10 : The User Agent source IP
- UA=SIP-UA 1.2.3.4 : The SIP User Agent header

In order to collect the full log related to a single call, it's necessary to "grep" the `/var/log/ngcp/kamailio-proxy.log` using the **ID=** string, for example:

```
# grep "364e4676776621034977934e055d19ea@127.0.0.1" /var/log/ngcp/kamailio-proxy.log
```

**17.10.2 Collecting SIP traces**

The Sipwise C5 platform provides several tools to collect SIP traces. It can be used Sipwise C5 `ngrep-sip` tool to collect SIP traces, for example to fetch traffic in text format from outbound and among load balancer, proxy and sems :

```
# ngrep-sip b
```

see the manual to know all the options:

```
# man ngrep-sip
```

The *ngrep* debian tool can be used in order to make a SIP trace and save it into a *.pcap* file :

```
# ngrep -s0 -Wbyline -d any -O /tmp/SIP_trace_file_name.pcap port 5062 or port 5060
```

The *sngrep* debian graphic tool as well can be used to visualize SIP trace and save them in a *.pcap* file :

```
# sngrep
```

## 18 Monitoring and Alerting

### 18.1 Internal Monitoring

#### 18.1.1 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.

#### 18.1.2 Sipwise C5 specific monitoring via ngcp-witnessd

The platform uses the internal *ngcp-witnessd* service to monitor Sipwise C5 specific metrics or system metrics currently not tracked by *telegraf*, including memory, process count, HA status, MTA, Kamailio, SIP and MySQL.

The gathered information is stored in *InfluxDB*, in the *ngcp* database.

#### 18.1.3 Monitoring data in InfluxDB

The platform uses *InfluxDB* as a time series database, to store most of the metrics collected in the system.

The monitoring data is used by the statistics dashboard powered by *Grafana*.

The monitoring data can also be accessed directly by various means; 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 *NGCP::InfluxDB::HTTP* perl module.

See [https://docs.influxdata.com/influxdb/v1.1/query\\_language/spec/](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 "measurement"`.

---

**Tip**

To get the list of tags for a specific measurement the following query can be used `SHOW TAG KEYS FROM "measurement"`, and for all the current tag values for a tag `SHOW TAG VALUES FROM "measurement" WITH KEY = "tag"`.

---

See Section [D.3](#) for detailed information about the list of data currently stored in the *InfluxDB ngcp* monitoring database.

## 18.2 Statistics Dashboard

The platform's administration interface (described in Section [6](#)) 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.

## A Basic Call Flows

### A.1 General Call Setup

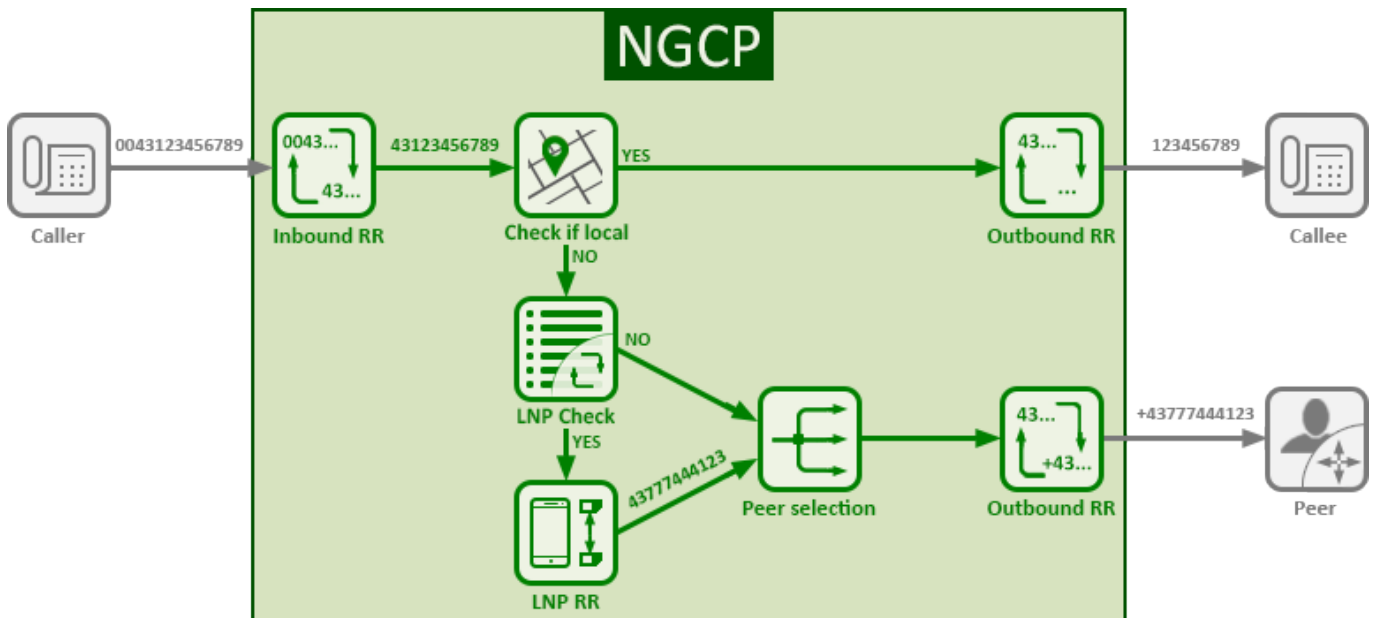


Figure 72: General Call Setup

Sipwise C5 performs the following checks when processing a call coming from a subscriber and terminated at a peer:

- Checks if the IP address where the request came from is in the list of trusted IP addresses. If yes, this IP address is taken as the identity for authentication. Otherwise, Sipwise C5 performs the digest authentication.
- When the subscriber is authorized to make the call, Sipwise C5 applies the Inbound Rewrite Rules for the caller and the callee assigned to the subscriber (if any). If there are no Rewrite Rules assigned to the subscriber, the ones assigned to the subscriber's domain are applied. On this stage the platform normalises the numbers from the subscriber's format to E.164.
- Matches the callee (called number) with local subscribers.
  - If it finds a matching subscriber, the call is routed internally. In this case, Sipwise C5 applies the Outbound Rewrite Rules associated with the callee (if any). If there are no Rewrite Rules assigned to the callee, the ones assigned to the callee's domain are applied.
  - If it does not find a matching subscriber, the call goes to a peer as described below.
- Queries the LNP database to find out if the number was ported or not. For details of LNP queries refer to the [Local Number Porting](#) chapter.
  - If it was ported, Sipwise C5 applies the LNP Rewrite Rules to the called number.
- Based on the priorities of peering groups and peering rules (see Section 6.6.2.3 for details), Sipwise C5 selects peering groups for call termination and defines their precedence.

- Within every peering group the weight of a peering server defines its probability to receive the call for termination. Thus, the bigger the weight of a server, the higher the probability that Sipwise C5 will send the call to it.
- Applies the Outbound Rewrite Rules for the caller and the callee assigned to a peering server when sending the call to it.

## A.2 Endpoint Registration

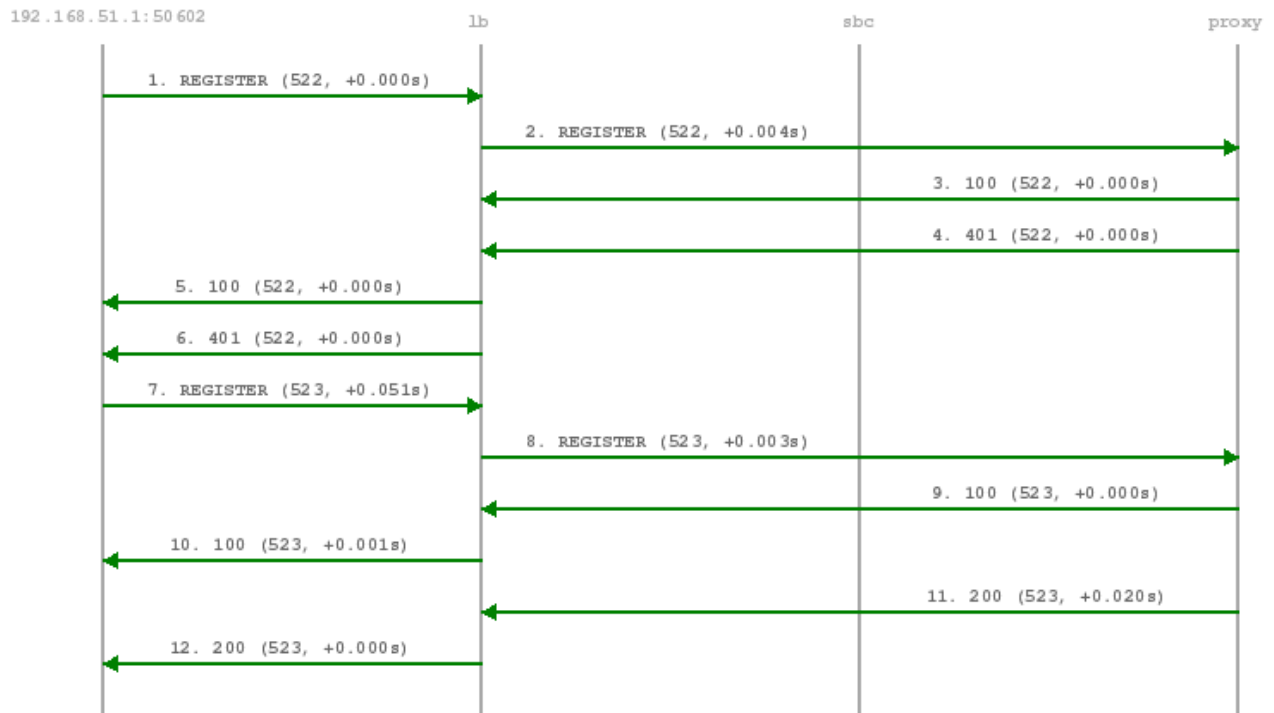


Figure 73: Registration Call-Flow

The subscriber endpoint starts sending a REGISTER request, which gets challenged by a 401. After calculating the response of the authentication challenge, it sends the REGISTER again, including the authentication response. The SIP proxy looks up the credentials of the subscriber in the database, does the same calculation, and if the result matches the one from the subscriber, the registration is granted.

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

If NAT is detected, the SIP proxy sends a `OPTION` message to the registered contact every 30 seconds, in order to keep the NAT binding on the NAT device open. Otherwise, for subsequent calls to this contact, Sipwise C5 wouldn't be able to reach the endpoint behind NAT (NAT devices usually drop a UDP binding after not receiving any traffic for ~30-60 seconds).

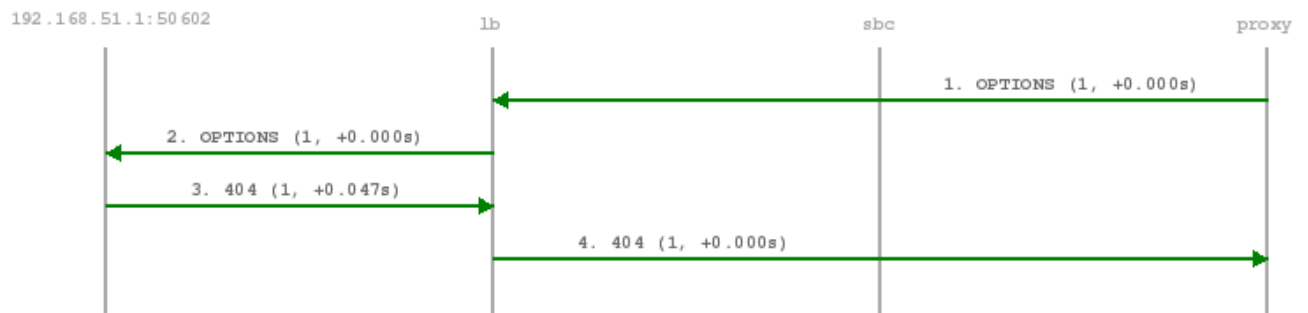


Figure 74: NAT-Ping Call-Flow

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





### A.3 Basic Call

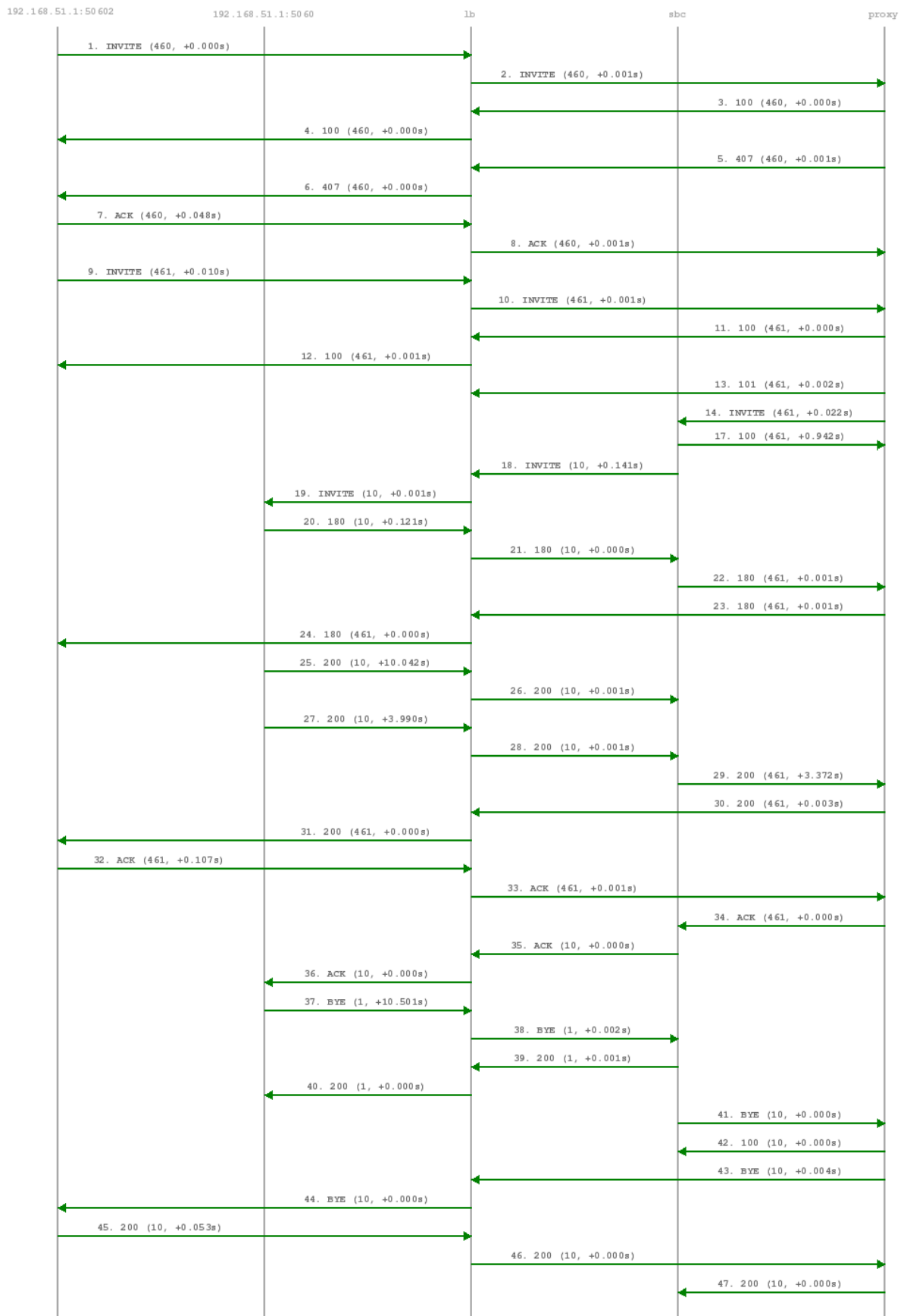


Figure 75: Basic Call Call-Flow

The calling party sends an INVITE (e.g. `sip:someuser@example.org`) via the SIP load-balancer to the SIP proxy. The proxy replies with an authorization challenge in the 407 response, and the calling party sends the INVITE again with authentication credentials. The SIP proxy checks if the called party is a local user. If it is, and if there is a registered contact found for this user, then (after various feature-related tasks for both the caller and the callee) the Request-URI is replaced by the URI of the registered contact (e.g. `sip:me@1.2.3.4:1234;transport=UDP`). If it's not a local user but a numeric user, a proper PSTN gateway is being selected by the SIP proxy, and the Request-URI is rewritten accordingly (e.g. `sip:+43123456789@2.3.4.5:5060`).

Once the proxy has finished working through the call features of both parties involved and has selected the final destination for the call, and - optionally - has invoked the Media Relay for this call, the INVITE is sent to the SIP B2BUA. The B2BUA creates a new INVITE message from scratch (using a new Call-ID and a new From-Tag), copies only various and explicitly allowed SIP headers from the old message to the new one, filters out unwanted media capabilities from the SDP body (e.g. to force audio calls to use G.711 as a codec) and then sends the new message via the SIP load-balancer to the called party.

SIP replies from the called party are passed through the elements back to the calling party (replacing various fields on the B2BUA to match the first call leg again). If a reply with an SDP body is received by the SIP proxy (e.g. a 183 or a 200), the Media Relay is invoked again to prepare the ports for the media stream.

Once the 200 is routed from the called party to the calling party, the media stream is fully negotiated, and the endpoints can start sending traffic to each other (either end-to-end or via the Media Relay). Upon reception of the 200, the SIP proxy writes a start record for the accounting process. The 200 is also acknowledged with an ACK message from the calling party to the called party, according to the SIP 3-way handshake.

Either of the parties can tear down the media session at any time by sending a BYE, which is passed through to the other party. Once the BYE reaches the SIP proxy, it instructs the Media Relay to close the media ports, and it writes a stop record for accounting purposes. Both the start- and the stop-records are picked up by the *ngcp-mediator* service in a regular interval and are converted into a Call Detail Record (CDR), which will be rated by the *ngcp-rate-o-mat* process and can be billed to the calling party.

## 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

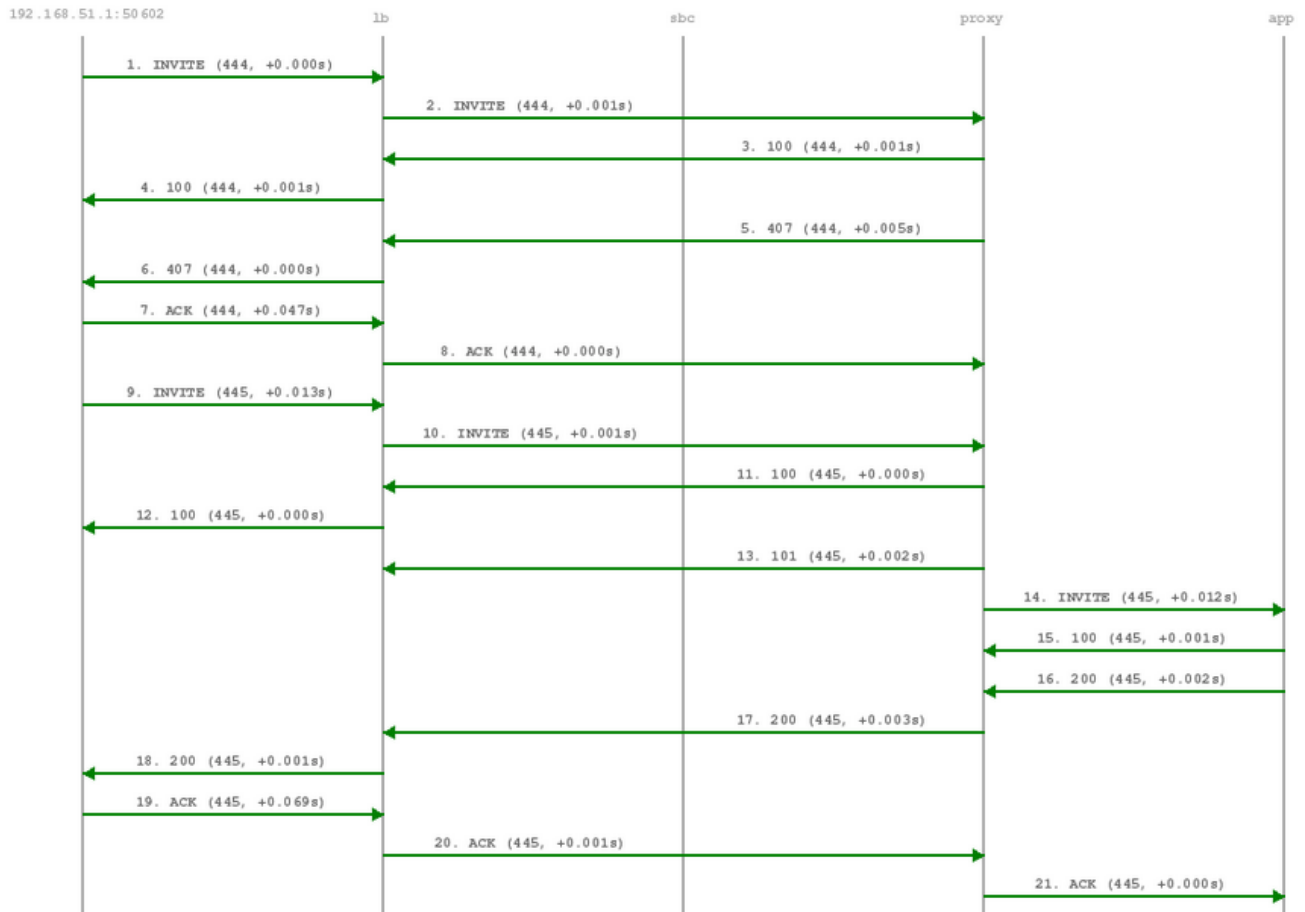


Figure 76: Voicebox Call-Flow

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 Sipwise C5 does not require your device to subscribe to the MWI service by sending a SUBSCRIBE (it would rather reject it). On the other hand, the endpoints need to accept unsolicited NOTIFY messages (that is, a NOTIFY without a valid subscription), otherwise the MWI service will not work with these endpoints.

---

## B Sipwise C5 configs overview

### B.1 config.yml Overview

`/etc/ngcp-config/config.yml` is the main configuration YAML file used by Sipwise C5. After every changes it need to run the command `ngcpcfg apply "my commit message"` to apply changes (followed by `ngcpcfg push` in the PRO version to apply changes to sp2). The following is a brief description of the main variables contained into `/etc/ngcp-config/config.yml` file.

#### B.1.1 apps

This section contains parameters for the additional applications that may be activated on Sipwise C5.

```
apps:
  malicious_call: no
```

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

#### B.1.2 asterisk

The following is the asterisk section:

```
asterisk:
  log:
    facility: local6
  rtp:
    maxport: 20000
    minport: 10000
  sip:
    bindport: 5070
    dtmfmode: rfc2833
  voicemail:
    enable: no
    fromstring: 'Voicemail server'
    greeting:
      busy_custom_greeting: '/home/user/file_no_extension'
      busy_overwrite_default: no
      busy_overwrite_subscriber: no
      unavail_custom_greeting: '/home/user/file_no_extension'
      unavail_overwrite_default: no
      unavail_overwrite_subscriber: no
```

```

mailbody: 'You have received a new message from ${VM_CALLERID} in voicebox ${VM_MAILBOX}
        } on ${VM_DATE}.'
mailsubject: '[Voicebox] New message ${VM_MSGNUM} in voicebox ${VM_MAILBOX}'
max_msg_length: 180
maxgreet: 60
maxmsg: 30
maxsilence: 0
min_msg_length: 3
normalize_match: '^00|\|+([1-9][0-9]+)$'
normalize_replace: '$1'
serveremail: voicebox@sip.sipwise.com

```

- `log.facility`: rsyslog facility for asterisk log, defined in `/etc/asterisk/logger.conf`.
- `rtp.maxport`: RTP maximum port used by asterisk.
- `rtp.minport`: RTP minimum port used by asterisk.
- `sip.bindport`: SIP asterisk internal bindport.
- `voicemail.greetings.*`: set the audio file path for voicemail custom unavailable/busy greetings
- `voicemail.mailbody`: Mail body for incoming voicemail.
- `voicemail.mailsubject`: Mail subject for incoming voicemail.
- `voicemail.max_msg_length`: Sets the maximum length of a voicemail message, in seconds.
- `voicemail.maxgreet`: Sets the maximum length of voicemail greetings, in seconds.
- `voicemail.maxmsg`: Sets the maximum number of messages that may be kept in any voicemail folder.
- `voicemail.min_msg_length`: Sets the minimum length of a voicemail message, in seconds.
- `voicemail.maxsilence`: Maxsilence defines how long Asterisk will wait for a contiguous period of silence before terminating an incoming call to voice mail. The default value is 0, which means the silence detector is disabled and the wait time is infinite.
- `voicemail.serveremail`: Provides the email address from which voicemail notifications should be sent.
- `voicemail.normalize_match`: Regular expression to match the From number for calls to voicebox.
- `voicemail.normalize_replace`: Replacement string to return, in order to match an existing voicebox.

### B.1.3 autoprov

The following is the autoprovisioning section:

```
autoprov:
  hardphone:
    skip_vendor_redirect: no
  server:
    bootstrap_port: 1445
    ca_certfile: '/etc/ngcp-config/ssl/client-auth-ca.crt'
    host: localhost
    port: 1444
    server_certfile: '/etc/ngcp-config/ssl/myserver.crt'
    server_keyfile: '/etc/ngcp-config/ssl/myserver.key'
    ssl_enabled: yes
  softphone:
    config_lockdown: 0
    webauth: 0
```

- `autoprov.skip_vendor_redirect`: Skip phone vendor redirection to the vendor provisioning web site.

#### B.1.4 backuptools

The following is the backup tools section:

```
backuptools:
  cdreexport_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: '/ngcp-data/backup/ngcp_backup'
  temp_backup_dir: '/ngcp-data/backup/ngcp_backup/tmp'
```

- `backuptools.cdreexport_backup.enable`: Enable backup of cdreexport (.csv) directory.
- `backuptools.etc_backup.enable`: Enable backup of /etc/\* directory.
- `backuptools.mail.address`: Destination email address for backup emails.



- `backuptools.mail.error_subject`: Subject for error emails.
- `backuptools.mail.log_subject`: Subject for daily backup report.
- `backuptools.mail.send_error`: Send daily backup error report.
- `backuptools.mail.send_log`: Send daily backup log report.
- `backuptools.mysql_backup.enable`: Enable daily mysql backup.
- `backuptools.mysql_backup.exclude_dbs`: exclude mysql databases from backup.
- `backuptools.rotate_days`: Number of days backup files should be kept. All files older than specified number of days are deleted from the storage directory.
- `backuptools.storage_dir`: Storage directory of backups.
- `backuptools.storage_group`: Name of the group that backup files should be owned by.
- `backuptools.storage_user`: Name of the user that backup files should be owned by.
- `backuptools.temp_backup_dir`: Temporary storage directory of backups.

### B.1.5 cdreexport

The following is the cdr export section:

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

- `cdreexport.daily_folder`: Set `yes` if you want to create a daily folder for CDRs under the configured path.
- `cdreexport.export_failed`: Export CDR for failed calls.
- `cdreexport.export_incoming`: Export CDR for incoming calls.
- `cdreexport.exportpath`: The path to store CDRs in `.csv` format.
- `cdreexport.full_names`: Use full names for CDRs instead of short ones.
- `cdreexport.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 cleanuptools

The following is the cleanup tools section:

```
cleanuptools:
  acc_cleanup_days: 90
  archive_targetdir: '/ngcp-data/backups/cdr'
  binlog_days: 15
  cdr_archive_months: 2
  cdr_backup_months: 2
  cdr_backup_retro: 3
  compress: gzip
  delete_old_cdr_files:
    enable: no
    max_age_days: 30
    paths:
      -
        max_age_days: ~
        path: '/home/jail/home/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]'
        remove_empty_directories: yes
        wildcard: yes
      -
        max_age_days: ~
        path: '/home/jail/home/cdrexpert/resellers/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]'
        remove_empty_directories: yes
        wildcard: yes
      -
        max_age_days: ~
        path: '/home/jail/home/cdrexpert/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`:
  - `enable`: Enable (`yes`) or disable (`no`) exported CDR cleanup.
  - `max_age_days`: Gives the expiration time of the exported CDR files in days. There is a general value which may be overridden by a local value provided at a specific path. The local value is valid for the particular path only.
  - `paths`: an array of path definitions
    - \* `path`: a path where CDR files are to be found and deleted; this may contain wildcard characters
    - \* `wildcard`: Enable (`yes`) or disable (`no`) using wildcards in the `path`
    - \* `remove_empty_directories`: Enable (`yes`) or disable (`no`) removing empty directories if those are found in the given `path`
    - \* `max_age_days`: the local expiration time value for files in the particular `path`
- `cleanuptools.sql_batch`: How many records to process within a single SQL statement.
- `cleanuptools.trash_cleanup_days`: Time after CDRs from `acc_trash` and `acc_backup` tables in `kamailio` database will be deleted.

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

### B.1.7 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.1.8 database

The following is the database section:

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

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

### B.1.9 faxserver

The following is the fax server section:

```
faxserver:
  enable: yes
  fail_attempts: '3'
  fail_retry_secs: '60'
  mail_from: 'Sipwise C5 FaxServer <voipfax@ngcp.sipwise.local>'
```

- `faxserver.enable`: *yes/no* to enable or disable `ngcp-faxserver` on the platform respectively.
- `faxserver.fail_attempts`: Amount of attempts to send a fax after which it is marked as *failed*.
- `faxserver.fail_retry_secs`: Amount of seconds to wait between "fail\_attempts".
- `faxserver.mail_from`: Sets the e-mail From Header for incoming fax.

### B.1.10 general

The following is the general section:

```
general:
  adminmail: adjust@example.org
  companyname: sipwise
  lang: en
  maintenance: no
  production: yes
  timezone: localtime
```

- `general.adminmail`: Email address used by `monit` to send notifications to.
- `general.companyname`: Label used in `SNMPd` configuration.
- `general.lang`: Sets sounds language (e.g. *de* for German)
- `general.production`: Label to hint self-check scripts about installation mode.
- `general.maintenance`: maintenance mode necessary for safe upgrades.
- `general.timezone`: Sipwise C5 Timezone

### B.1.11 heartbeat

The following is the heartbeat section:

```
heartbeat:
  hb_watchdog:
    action_max: 5
    enable: yes
    interval: 10
    transition_max: 10
```

- `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.

### B.1.12 intercept

The following is the legal intercept section:

```
intercept:
  enable: no
```

- `intercept.enable`: Enable ngcp-voisniff for Lawful Interception (additional Sipwise C5 module).

### B.1.13 kamailio

The following is the kamailio section:

```
kamailio:
  lb:
    cfgt: no
  debug:
    enable: no
  modules:
    - level: '1'
      name: core
    - level: '3'
      name: xlog
```

```
debug_level: '1'
debug_uri:
  enable: no
  redis_db: 27
  htable_idx_size: 4
dns:
  dns_sctp_pref: 1
  dns_tcp_pref: 1
  dns_tls_pref: 1
  dns_try_naptr: no
  dns_udp_pref: 1
  use_dns_cache: on
external_sbc: []
extra_sockets: ~
max_forwards: '70'
mem_log: '1'
mem_summary: '12'
max_inv_lifetime: '180000'
natatest_exception_ips:
- 1.2.3.4
- 5.6.7.8
pkg_mem: '16'
port: '5060'
remove_isup_body_from_replies: no
sdp_line_filter:
  enable: no
  remove_line_startswith: []
security:
  dos_ban_enable: yes
  dos_ban_time: '300'
  dos_reqs_density_per_unit: '50'
  dos_sampling_time_unit: '5'
  dos_whitelisted_ips: []
  dos_whitelisted_subnets: []
  failed_auth_attempts: '3'
  failed_auth_ban_enable: yes
  failed_auth_ban_time: '3600'
topoh:
  enable: no
  mask_callid: no
  mask_ip: 127.0.0.8
topos:
  enable: no
  redis_db: 24
shm_mem: '64'
skip_contact_alias_for_ua_when_tcp:
  enable: no
  user_agent_patterns: []
```

```
start: yes
strict_routing_safe: no
syslog_options: yes
tcp_children: 1
tcp_max_connections: '2048'
tls:
  enable: no
  port: '5061'
  sslcertfile: /etc/ngcp-config/ssl/myserver.crt
  sslcertkeyfile: /etc/ngcp-config/ssl/myserver.key
udp_children: 1
proxy:
  allow_cf_to_itself: no
  allow_info_method: no
  allow_msg_method: no
  allow_peer_relay: no
  allow_refer_method: no
  always_anonymize_from_user: no
  authenticate_bye: no
  block_useragents:
    action: reject
    enable: no
    mode: blacklist
    ua_patterns: []
  cf_depth_limit: '10'
  cfgt: no
  check_prev_forwarder_as_upn: no
  children: 1
  decode_utu_header: no
  debug:
    enable: no
    modules:
      - level: '1'
        name: core
      - level: '3'
        name: xlog
  debug_level: '1'
  default_expires: '3600'
  default_expires_range: '30'
  dlg_timeout: '43200'
  early_rejects:
    block_admin:
      announce_code: '403'
      announce_reason: Blocked by Admin
    block_callee:
      announce_code: '403'
      announce_reason: Blocked by Callee
    block_caller:
```

```
announce_code: '403'
announce_reason: Blocked by Caller
block_contract:
  announce_code: '403'
  announce_reason: Blocked by Contract
block_in:
  announce_code: '403'
  announce_reason: Block in
block_out:
  announce_code: '403'
  announce_reason: Blocked out
block_override_pin_wrong:
  announce_code: '403'
  announce_reason: Incorrect Override PIN
callee_busy:
  announce_code: '486'
  announce_reason: Busy Here
callee_offline:
  announce_code: '480'
  announce_reason: Offline
callee_tmp_unavailable:
  announce_code: '480'
  announce_reason: Temporarily Unavailable
callee_tmp_unavailable_gp:
  announce_code: '480'
  announce_reason: Unavailable
callee_tmp_unavailable_tm:
  announce_code: '408'
  announce_reason: Request Timeout
callee_unknown:
  announce_code: '404'
  announce_reason: Not Found
cf_loop:
  announce_code: '480'
  announce_reason: Unavailable
emergency_invalid:
  announce_code: '404'
  announce_reason: Emergency code not available in this region
emergency_unsupported:
  announce_code: '403'
  announce_reason: Emergency Calls Not Supported
invalid_speeddial:
  announce_code: '484'
  announce_reason: Speed-Dial slot empty
locked_in:
  announce_code: '403'
  announce_reason: Callee locked
locked_out:
```



```
announce_code: '403'
announce_reason: Caller locked
max_calls_in:
  announce_code: '486'
  announce_reason: Busy
max_calls_out:
  announce_code: '403'
  announce_reason: Maximum parallel calls exceeded
no_credit:
  announce_code: '402'
  announce_reason: Insufficient Credit
peering_unavailable:
  announce_code: '503'
  announce_reason: PSTN Termination Currently Unavailable
reject_vsc:
  announce_code: '403'
  announce_reason: VSC Forbidden
relaying_denied:
  announce_code: '403'
  announce_reason: Relaying Denied
unauth_caller_ip:
  announce_code: '403'
  announce_reason: Unauthorized IP detected
emergency_priorization:
  enable: no
  register_fake_200: yes
  register_fake_expires: '3600'
  reject_code: '503'
  reject_reason: Temporary Unavailable
  retry_after: '3600'
enum_suffix: e164.arpa.
expires_range: '30'
filter_100rel_from_supported: no
filter_failover_response: 408|500|503
foreign_domain_via_peer: no
fritzbox:
  enable: no
  prefixes:
    - 0$avp(caller_ac)
    - $avp(caller_cc)$avp(caller_ac)
    - \+$avp(caller_cc)$avp(caller_ac)
    - 00$avp(caller_cc)$avp(caller_ac)
  special_numbers:
    - '112'
    - '110'
    - 118[0-9]{2}
ignore_auth_realm: no
ignore_subscriber_allowed_clis: no
```

```
keep_original_to: no
latency_limit_action: '100'
latency_limit_db: '500'
latency_log_level: '1'
latency_runtime_action: 1000
lnp:
  add_reply_headers:
    enable: no
    number: P-NGCP-LNP-Number
    status: P-NGCP-LNP-Status
  api:
    add_caller_cc_to_lnp_dst: no
    invalid_lnp_routing_codes:
      - ^EE00
      - ^DD00
    keepalive_interval: '3'
    lnp_request_blacklist: []
    lnp_request_whitelist: []
    port: '8991'
    reply_error_on_lnp_failure: no
    request_timeout: '1000'
    server: localhost
    tcap_field_fci: end.components.0.invoke.parameter
    tcap_field_lnp: ConnectArg.destinationRoutingAddress.0
    tcap_field_opcode: end.components.0.invoke.opCode
  enable: no
  execute_ncos_block_out_before_lnp: no
  skip_callee_lnp_lookup_from_any_peer: no
  strictly_check_ncos: no
  type: api
lookup_peer_destination_domain_for_pbx: no
loop_detection:
  enable: no
  expire: '1'
  max: '5'
max_expires: '43200'
max_gw_lcr: '128'
max_registrations_per_subscriber: '5'
mem_log: '1'
mem_summary: '12'
min_expires: '60'
nathelper:
  sipping_from: sip:pinger@sipwise.local
nathelper_dbro: no
natping_interval: '30'
natping_processes: 1
nonce_expire: '300'
pbx:
```

```
hunt_display_fallback_format: '[H %s]'  
hunt_display_fallback_indicator: $var(cloud_pbx_hg_ext)  
hunt_display_format: '[H %s]'  
hunt_display_indicator: $var(cloud_pbx_hg_displayname)  
hunt_display_maxlength: 8  
ignore_cf_when_hunting: no  
skip_busy_hg_members:  
  enable: no  
  redis_key_name: totaluser  
peer_probe:  
  available_treshold: '1'  
  enable: yes  
  from_uri_domain: probe.ngcp.local  
  from_uri_user: ping  
  interval: '10'  
  method: OPTIONS  
  reply_codes: class=2;class=3;code=403;code=404;code=405  
  timeout: '5'  
  unavailable_treshold: '1'  
perform_peer_failover_on_tm_timeout: yes  
perform_peer_lcr: no  
pkg_mem: '32'  
port: '5062'  
presence:  
  enable: yes  
  max_expires: '3600'  
  reginfo_domain: example.org  
proxy_lookup: no  
push:  
  apns_alert: New call  
  apns_sound: incoming_call.xaf  
report_mos: yes  
set_ruri_to_peer_auth_realm: no  
shm_mem: '125'  
start: yes  
store_recentcalls: no  
syslog_options: yes  
tcp_children: 1  
tm:  
  fr_inv_timer: '180000'  
  fr_timer: '9000'  
  max_inv_lifetime: '180000'  
treat_600_as_busy: yes  
use_enum: no  
usrloc_dbmode: '1'  
voicebox_first_caller_cli: yes  
xfer_other_party_from: no
```

- `kamailio.lb.cfgt`: Enable/disable unit test config file execution tracing.
- `kamailio.lb.debug.enable`: Enable per-module debug options.
- `kamailio.lb.debug.modules`: List of modules to be traced with respective debug level.
- `kamailio.lb.debug.uri.enable`: Enable/disable sending SIP messages From/To specific subscriber to an inactive proxy node in order to debug/trace calls. Only makes sense on Sipwise C5 CARRIER appliance environment.
- `kamailio.lb.debug.uri.redis_db`: A number of internal Redis DB used by `htable` module to keep the subscribers values
- `kamailio.lb.debug.uri.htable_idx_size`: number to control how many slots (buckets) to create for the hash table (  $2^{\text{size}}$  ). See [kamailio htable](#) docs for details.
- `kamailio.lb.debug.level`: Default debug level for `kamailio-lb`.
- `kamailio.lb.dns.use_dns_cache`: Enable/disable use of internal DNS cache.
- `kamailio.lb.dns.dns_udp_pref`: Set preference for each protocol when doing NAPTR lookups. In order to use remote site preferences set all `dns_*_pref` to the same positive value (e.g. `dns_udp_pref=1`, `dns_tcp_pref=1`, `dns_tls_pref=1`, `dns_sctp_pref=1`). To completely ignore NAPTR records for a specific protocol, set the corresponding protocol preference to `-1`.
- `kamailio.lb.dns.dns_tcp_pref`: See above.
- `kamailio.lb.dns.dns_tls_pref`: See above.
- `kamailio.lb.dns.dns_sctp_pref`: See above.
- `kamailio.lb.dns.dns_try_naptr`: Enable NAPTR support according to RFC 3263.
- `kamailio.lb.external_sbc`: SIP URI of external SBC used in the Via Route option of peering server.
- `kamailio.lb.extra_sockets`: Add here extra sockets for Load Balancer.
- `kamailio.lb.max_forwards`: Set the value for the Max Forwards SIP header for outgoing messages.
- `kamailio.lb.mem_log`: Specifies on which log level the memory statistics will be logged.
- `kamailio.lb.mem_summary`: Parameter to control printing of memory debugging information on exit or SIGUSR1 to log.
- `kamailio.lb.max_inv_lifetime`: Set INVITE transaction timeout per the whole transaction if no final reply for an INVITE arrives after a provisional message was received (whole transaction ringing timeout). It has to be equals or greater than `kamailio.proxy.tm.fr_inv_timer`.
- `kamailio.lb.nattest_exception_ips`: List of IPs that don't need the NAT test.
- `kamailio.lb.shm_mem`: Shared memory used by Kamailio Load Balancer.
- `kamailio.lb.pkg_mem`: PKG memory used by Kamailio Load Balancer.
- `kamailio.lb.port`: Default listen port.
- `kamailio.lb.remove_isup_body_from_replies`: Enable/disable stripping of ISUP part from the message body.
- `kamailio.lb.sdp_line_filter.enable`: Enable/Disable filter of SDP lines in all the SIP messages.

- `kamailio.lb.sdp_line_filter.remove_line_startswith`: List of the SDP lines that should be removed. Attention: it removes all SDP attribute lines beginning with the listed strings in all media streams.
- `kamailio.lb.security.dos_ban_enable`: Enable/Disable DoS Ban.
- `kamailio.lb.security.dos_ban_time`: Sets the ban time.
- `kamailio.lb.security.dos_reqs_density_per_unit`: Sets the requests density per unit (if we receive more than \* `lb.dos_reqs_density_per_unit` within `dos_sampling_time_unit` the user will be banned).
- `kamailio.lb.security.dos_sampling_time_unit`: Sets the DoS unit time.
- `kamailio.lb.security.dos_whitelisted_ips`: Write here the whitelisted IPs.
- `kamailio.lb.security.dos_whitelisted_subnets`: Write here the whitelisted IP subnets.
- `kamailio.lb.security.failed_auth_attempts`: Sets how many authentication attempts allowed before ban.
- `kamailio.lb.security.failed_auth_ban_enable`: Enable/Disable authentication ban.
- `kamailio.lb.security.failed_auth_ban_time`: Sets how long a user/IP has be banned.
- `kamailio.lb.topoh.enable`: Enable topology masking module (see the [Topology Masking Mechanism](#) subchapter for a detailed description).
- `kamailio.lb.topoh.mask_callid`: if set to `yes`, the SIP Call-ID header will also be encoded.
- `kamailio.lb.topoh.mask_ip`: an IP address that will be used to create valid SIP URIs, after encoding the real/original header content.
- `kamailio.lb.topos.enable`: Enable topology hiding module (see the [Topology Hiding Mechanism](#) subchapter for a detailed description).
- `kamailio.lb.topos.redis_db`: A number of internal Redis DB used by the topology hiding module.
- `kamailio.lb.start`: Enable/disable kamailio-lb service.
- `kamailio.lb.strict_routing_safe`: Enable strict routing handle feature.
- `kamailio.lb.syslog_options`: Enable/disable logging of SIP OPTIONS messages to `kamailio-options-lb.log`.
- `kamailio.lb.tcp_children`: Number of TCP worker processes.
- `kamailio.lb.tcp_max_connections`: Maximum number of open TCP connections.
- `kamailio.lb.tls.enable`: Enable TLS socket.
- `kamailio.lb.tls.port`: Set TLS listening port.
- `kamailio.lb.tls.sslcertificate`: Path for the SSL certificate.
- `kamailio.lb.tls.sslcertkeyfile`: Path for the SSL key file.
- `kamailio.lb.udp_children`: Number of UDP worker processes.
- `kamailio.proxy.allow_cf_to_itself`: Specify whether or not a Call Forward to the same subscriber (main number to an alias or viceversa) is allowed. To stop the CF loop a source number or a b-number have to be defined in the CF configuration.

- `kamailio.proxy.allow_info_method`: Allow INFO method.
- `kamailio.proxy.allow_msg_method`: Allow MESSAGE method.
- `kamailio.proxy.allow_peer_relay`: Allow peer relay. Call coming from a peer that doesn't match a local subscriber will try to go out again, matching the peering rules.
- `kamailio.proxy.allow_refer_method`: Allow REFER method. Enable it with caution.
- `kamailio.proxy.always_anonymize_from_user`: Enable anonymization of full From URI (as opposed to just From Display-name part by default), has same effect as enabling the preference `anonymize_from_user` for all peers.
- `kamailio.proxy.authenticate_bye`: Enable BYE authentication.
- `kamailio.proxy.block_useragents.action`: one of [`drop`, `reject`] - Whether to silently drop the request from matching User-Agent or reject with a 403 message.
- `kamailio.proxy.block_useragents.enable`: Enable/disable the User-Agent blocking.
- `kamailio.proxy.block_useragents.mode`: one of [`whitelist`, `blacklist`] - Sets the mode of `ua_patterns` list evaluation (`whitelist`: block requests coming from all but listed User-Agents, `blacklist`: block requests from all listed User-Agents).
- `kamailio.proxy.block_useragents.ua_patterns`: List of User-Agent string patterns that trigger the block action.
- `kamailio.proxy.cf_depth_limit`: CF loop detector. How many CF loops are allowed before drop the call.
- `kamailio.proxy.cfgt`: Enable/disable unit test config file execution tracing.
- `kamailio.proxy.check_prev_forwarder_as_upn`: Enable/disable validation of the forwarder's number taken from the `Diversion` or `History-Info` header.
- `kamailio.proxy.children`: Number of UDP worker processes.
- `kamailio.proxy.decode_uu_header`: Default `no`. If set to `yes`, the content of the User-to-User field received in 200Ok is decoded and saved in a dedicated field of the ACC records. The decoding consists in few steps: discard everything after the first occurrence of `,`; remove the initial `04`, hex decode the remaining part.
- `kamailio.proxy.debug.enable`: Enable per-module debug options.
- `kamailio.proxy.debug.modules`: List of modules to be traced with respective debug level.
- `kamailio.proxy.debug_level`: Default debug level for `kamailio-proxy`.
- `kamailio.proxy.default_expires`: Default expires value in seconds for a new registration (for REGISTER messages that contains neither Expires HFs nor expires contact parameters).
- `kamailio.proxy.default_expires_range`: This parameter specifies that the expiry used for the registration should be randomly chosen in a range given by `default_expires +/- default_expires_range` percent. For instance, if `default_expires` is 1200 seconds and `default_expires_range` is 50, the expiry is randomly chosen between [600,1800] seconds. If set to `0`, `default_expires` is left unmodified.
- `kamailio.proxy.dlg_timeout`: Dialog timeout in seconds (by default 43200 sec - 12 hours).
- `kamailio.proxy.early_rejects`: Customize here the response codes and sound prompts for various reject scenarios. See the subchapter [Configuring Early Reject Sound Sets](#) for a detailed description.

- `kamailio.proxy.emergency_prioritization.enable`: Enable an emergency mode support.
- `kamailio.proxy.emergency_prioritization.register_fake_200`: When enabled, generates a fake 200 response to REGISTER from non-prioritized subscriber in emergency mode.
- `kamailio.proxy.emergency_prioritization.register_fake_expires`: Expires value for the fake 200 response to REGISTER.
- `kamailio.proxy.emergency_prioritization.reject_code`: Reject code for the non-emergency request.
- `kamailio.proxy.emergency_prioritization.reject_reason`: Reject reason for the non-emergency request.
- `kamailio.proxy.emergency_prioritization.retry_after`: Retry-After value when rejecting the non-emergency request.

---

**Tip**

In order to learn about details of *emergency prioritization* function of NGCP please refer to Section 7.8 part of the handbook.

---

- `kamailio.proxy.enum_suffix`: Sets ENUM suffix - don't forget . (dot).
- `kamailio.proxy.expires_range`: Set randomization of expires for REGISTER messages (similar to `default_expires_range` but applies to received expires value).
- `kamailio.proxy.filter_100rel_from_supported`: Enable filtering of *100rel* from Supported header, to disable PRACK.
- `kamailio.proxy.filter_failover_response`: Specify the list of SIP responses that trigger a failover on the next available peering server.
- `kamailio.proxy.foreign_domain_via_peer`: Enable/disable of routing of calls to foreign SIP URI via peering servers.
- `kamailio.proxy.fritzbox.enable`: Enable detection for Fritzbox special numbers. Ex. Fritzbox add some prefix to emergency numbers.
- `kamailio.proxy.fritzbox.prefixes`: Fritzbox prefixes to check. Ex. `0$avp(caller_ac)`
- `kamailio.proxy.fritzbox.special_numbers`: Specifies Fritzbox special number patterns. They will be checked with the prefixes defined. Ex. `112`, so the performed check will be `sip:0$avp(caller_ac)112@` if prefix is `0$avp(caller_ac)`
- `kamailio.proxy.ignore_auth_realm`: Ignore SIP authentication realm.
- `kamailio.proxy.ignore_subscriber_allowed_clis`: Set to *yes* to ignore the subscriber's `allowed_clis` preference so that the User-Provided CLI is only checked against customer's `allowed_clis` preference.
- `kamailio.proxy.latency_limit_action`: Limit of runtime in ms for config actions. If a config action executed by `cfg` interpreter takes longer than this value, a message is printed in the logs.
- `kamailio.proxy.latency_limit_db`: Limit of runtime in ms for DB queries. If a DB operation takes longer than this value, a warning is printed in the logs.
- `kamailio.proxy.latency_log_level`: Log level to print the messages related to latency. Default is 1 (INFO).
- `kamailio.proxy.latency_runtime_action`: Limit of runtime in ms for SIP message processing cycle. If the SIP message processing takes longer than this value, a warning is printed in the logs.

- `kamailio.proxy.inp.keep_original_to`: Not used now.
- `kamailio.proxy.inp.add_reply_headers.enable`: Enable/disable dedicated headers to be added after LNP lookup.
- `kamailio.proxy.inp.add_reply_headers.number`: Name of the header that will contain the LNP number.
- `kamailio.proxy.inp.add_reply_headers.status`: Name of the header that will contain the LNP return code (200 if OK, 500/480/... if an error/timeout is occurred).
- `kamailio.proxy.inp.api.add_caller_cc_to_inp_dst`: Enable/disable adding of caller country code to LNP routing number of the result (*no* by default, LNP result in E.164 format is assumed).
- `kamailio.proxy.inp.api.invalid_inp_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.inp.api.keepalive_interval`: Not used now.
- `kamailio.proxy.inp.api.inp_request_whitelist` [only for `api` type]: list of matching patterns of called numbers for which LNP lookup must be done.
- `kamailio.proxy.inp.api.inp_request_blacklist` [only for `api` type]: list of matching patterns of called numbers for which LNP lookup must not be done.
- `kamailio.proxy.inp.api.port`: Not used now.
- `kamailio.proxy.inp.api.reply_error_on_inp_failure`: Specifies whether platform should drop the call in case of LNP API server failure or continue routing the call to the original callee without LNP.
- `kamailio.proxy.inp.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.inp.api.server`: Not used now.
- `kamailio.proxy.inp.api.tcap_field_fci`: path of the FCI INFO in the received tcap message
- `kamailio.proxy.inp.api.tcap_field_inp`: path of the LNP NUMBER in the received tcap/inap message
- `kamailio.proxy.inp.api.tcap_field_opcode`: path of the FCI OPCODE in the received tcap message
- `kamailio.proxy.inp.enable`: Enable/disable LNP (local number portability) lookup during call setup.
- `kamailio.proxy.inp.execute_ncos_block_out_before_inp`: if set to *yes*, the NCOS and BLOCK\_OUT checks will be executed before the LNP lookup. Default is *no*, therefore the check are done after the LNP evaluation and rewriting.
- `kamailio.proxy.inp.skip_callee_inp_lookup_from_any_peer`: if set to *yes*, the destination LNP lookup is skipped (has same effect as enabling preference `skip_callee_inp_lookup_from_any_peer` for all peers).
- `kamailio.proxy.inp.strictly_check_ncos`: specify whether the NCOS LNP should be evaluated even if the LNP lookup was not previously executed or if it didn't return any occurrence. If set to *yes*, a whitelist NCOS will fail if the LNP lookup doesn't return any match. The parameter has no impact on blacklist NCOS.
- `kamailio.proxy.inp.type`: method of LNP lookup; valid values are: `local` (local LNP database) and `api` (LNP lookup through external gateways). *PLEASE NOTE*: the `api` type of LNP lookup is only available for Sipwise C5 PRO / CARRIER installations.



- `kamailio.proxy.lookup_peer_destination_domain_for_pbx`: one of [yes, no, peer\_host\_name] - Sets the content of destination\_domain CDR field for calls between CloudPBX subscribers. In case of *no* this field contains name of CloudPBX domain; *yes*: peer destination domain; *peer\_host\_name*: human-readable name of the peering server.
- `kamailio.proxy.loop_detection.enable`: Enable the SIP loop detection based on the combination of SIP-URI, To and From header URIs.
- `kamailio.proxy.loop_detection.expire`: Sampling interval in seconds for the incoming INVITE requests (by default 1 sec).
- `kamailio.proxy.loop_detection.max`: Maximum allowed number of SIP requests with the same SIP-URI, To and From header URIs within sampling interval. Requests in excess of this limit will be rejected with 482 Loop Detected response.
- `kamailio.proxy.max_expires`: Sets the maximum expires in seconds for registration. If set to 0, the check is disabled.
- `kamailio.proxy.max_gw_lcr`: Defines the maximum number of gateways in lcr\_gw table
- `kamailio.proxy.max_registrations_per_subscriber`: Sets the maximum registration per subscribers.
- `kamailio.proxy.mem_log`: Specifies on which log level the memory statistics will be logged.
- `kamailio.proxy.mem_summary`: Parameter to control printing of memory debugging information on exit or SIGUSR1 to log.
- `kamailio.proxy.min_expires`: Sets the minimum expires in seconds for registration. If set to 0, the check is disabled.
- `kamailio.proxy.nathelper.sipping_from`: Set the From header in OPTIONS NAT ping.
- `kamailio.proxy.nathelper_dbro`: Default is "no". This will be "yes" on CARRIER in order to activate the use of a read-only connection using LOCAL\_URL
- `kamailio.proxy.natping_interval`: Sets the NAT ping interval in seconds.
- `kamailio.proxy.natping_processes`: Set the number of NAT ping worker processes.
- `kamailio.proxy.nonce_expire`: Nonce expire time in seconds.
- `kamailio.proxy.pbx.hunt_display_fallback_format`: Default is [H %s]. Sets the format of the hunt group indicator that is sent as initial part of the From Display Name when subscriber is called as a member of PBX hunt group if the preferred format defined by the `hunt_display_format` and `hunt_display_indicator` can not be used (as in the case of not provisioned subscriber settings). The %s part is replaced with the value of the `hunt_display_fallback_indicator` variable.
- `kamailio.proxy.pbx.hunt_display_fallback_indicator`: The internal kamailio variable that sets the number or extension of the hunt group. Default is `$var (cloud_pbx_hg_ext)` which is populated during call routing with the extension of the hunt group.
- `kamailio.proxy.pbx.hunt_display_format`: Default is [H %s]. Sets the format of hunt group indicator that is sent as initial part of the From Display Name when subscriber is called as a member of PBX hunt group. This is the preferred (default) indicator format with Display Name, where the %s part is replaced with the value of the `hunt_display_indicator` variable.
- `kamailio.proxy.pbx.hunt_display_indicator`: The internal kamailio variable that contains the preferred identifier of the hunt group. Default is `$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.pbx.skip_busy_hg_members.enable`: Default is *no*. Whether to skip the subscribers that have busy status when routing the calls to huntgroups.
- `kamailio.proxy.pbx.skip_busy_hg_members.redis_key_name`: one of [`totaluser`, `activeuser`] - Sets the internal redis key name that contains the number of active calls for the user.
- `kamailio.proxy.peer_probe.enable`: Enable the peer probing, must be also checked per individual peer in the panel/API.
- `kamailio.proxy.peer_probe.interval`: Peer probe interval in seconds.
- `kamailio.proxy.peer_probe.timeout`: Peer probe response wait timeout in seconds.
- `kamailio.proxy.peer_probe.reply_codes`: Defines the response codes that are considered successful response to the configured probe request, e.g. `class=2; class=3; code=403; code=404; code=405`, with class defining a code range.
- `kamailio.proxy.peer_probe.unavailable_treshold`: Defines after how many failed probes a peer is considered unavailable.
- `kamailio.proxy.peer_probe.available_treshold`: Defines after how many successful probes a peer is considered available.
- `kamailio.proxy.peer_probe.from_uri_user`: From-userpart for the probe requests.
- `kamailio.proxy.peer_probe.from_uri_domain`: From-hostpart for the probe requests.
- `kamailio.proxy.peer_probe.method`: [OPTIONS|INFO] - Request method for probe request.

---

**Tip**

You can find more information about peer probing configuration in Section [7.12.2](#) of the handbook.

---

- `kamailio.proxy.perform_peer_failover_on_tm_timeout`: Specifies the failover behavior when maximum ring timeout (`fr_inv_timer`) has been reached. In case it is set to *yes*: failover to the next peer if any; in case of *no* stop trying other peers.
- `kamailio.proxy.perform_peer_lcr`: Enable/Disable Least Cost Routing based on peering fees.
- `kamailio.proxy.pkg_mem`: PKG memory used by Kamailio Proxy.
- `kamailio.proxy.shm_mem`: Shared memory used by Kamailio Proxy.
- `kamailio.proxy.port`: SIP listening port.
- `kamailio.proxy.presence.enable`: Enable/disable presence feature
- `kamailio.proxy.presence.max_expires`: Sets the maximum expires value for PUBLISH/SUBSCRIBE message. Defines expiration of the presentity record.
- `kamailio.proxy.presence.reginfo_domain`: Set FQDN of Sipwise C5 domain used in callback for mobile push.
- `kamailio.proxy.push.apns_alert`: Set the content of *alert* field towards APNS.
- `kamailio.proxy.push.apns_sound`: Set the content of *sound* field towards APNS.

- `kamailio.proxy.report_mos`: Enable MOS reporting in the log file.
- `kamailio.proxy.set_ruri_to_peer_auth_realm`: Set R-URI using peer auth realm.
- `kamailio.proxy.start`: Enable/disable kamailio-proxy service.
- `kamailio.proxy.store_recentcalls`: Store recent calls to redis (used by Malicious Call Identification application and VSCs related to recent calls redial).
- `kamailio.proxy.syslog_options`: Enable/disable logging of SIP OPTIONS messages to `kamailio-options-proxy.log`.
- `kamailio.proxy.tcp_children`: Number of TCP worker processes.
- `kamailio.proxy.tm.fr_inv_timer`: Set INVITE transaction timeout per branch if no final reply for an INVITE arrives after a provisional message was received (branch ringing timeout).
- `kamailio.proxy.tm.fr_timer`: Set INVITE transaction timeout if the destination is not responding with provisional response message.
- `kamailio.proxy.tm.max_inv_lifetime`: Set INVITE transaction timeout per the whole transaction if no final reply for an INVITE arrives after a provisional message was received (whole transaction ringing timeout). It has to be equals or greater than `kamailio.proxy.tm.fr_inv_timer`.
- `kamailio.proxy.treat_600_as_busy`: Enable the 6xx response handling according to RFC3261. When enabled, the 6xx response should stop the serial forking. Also, CFB will be triggered or busy prompt played as in case of 486 Busy response.
- `kamailio.proxy.use_enum`: Enable/Disable ENUM feature.
- `kamailio.proxy.usrloc_dbmode`: Set the mode of database usage for persistent contact storage.
- `kamailio.proxy.voicebox_first_caller_cli`: When enabled the previous forwarder's CLI will be used as caller CLI in case of chained Call Forwards.
- `kamailio.proxy.xfer_other_party_from`: If set to *yes* transferred calls will have the number of the transferred party in the From header. Default is *no*, thus transferred calls have the number of the transferrer party in the From header.

#### B.1.14 ngcp-mediator

The following is the `ngcp-mediator` section:

```
mediator:  
  interval: 10
```

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

### B.1.15 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.1.16 monitoring

The following is the check tools section:

```
monitoring:
  interval: 10
  retrospect_interval: 30
  threshold:
    cpu_idle_min: '0.1'
    disk_used_max: '0.9'
    kamilio_lb_shmem_min: '1048576'
    kamilio_proxy_shmem_min: '1048576'
    load_long_max: '2'
    load_medium_max: '2'
    load_short_max: '3'
    mem_used_max: 0.98
    mta_queue_len_max: '15'
    sip_responsiveness_max: '15'
    sslcert_timetoexpiry: '30'
    sslcert_whitelist: []
    swap_free_min: 0.02
```

- `monitoring.interval`: The number of seconds between each data gathering iteration.
- `monitoring.retrospect_interval`: The number of seconds to look into the past, when checking for the last value for a data point.
- `monitoring.threshold.cpu_idle_min`: Sets the minimum value for CPU usage (0.1 means 10%).

- `monitoring.threshold.disk_used_max`: Sets the maximum value for DISK usage (0.9 means 90%).
- `monitoring.threshold.kamailio_lb_shmem_min`: Sets the minimum value for Kamailio lb share memory usage.
- `monitoring.threshold.kamailio_proxy_shmem_min`: Sets the minimum value for Kamailio proxy share memory usage.
- `monitoring.threshold.load_long_max/load_long_max/load_short_max`: Max values for load (long, short, medium term).
- `monitoring.threshold.mem_used_max`: Sets the maximum value for memory usage (0.7 means 70%).
- `monitoring.threshold.mta_queue_len_max`: Sets the maximum value for the MTA queue length.
- `monitoring.threshold.sip_responsiveness_max`: Sets the maximum SIP responsiveness time timeout for the SIP options.
- `monitoring.threshold.sslcert_timetoexpiry`: Sets the number of days before a SSL certificate expiry starts to warn.
- `monitoring.threshold.sslcert_whitelist`: Sets a list of SSL certificate fingerprints to whitelist from the expiry check.
- `monitoring.threshold.swap_free_min`: Sets the minimum value for free swap (0.5 means 50%).

### B.1.17 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.1.18 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.1.19 ossbss

The following is the ossbss section:

```
ossbss:
  apache:
    port: 2443
    proxylisten: 1080
    restapi:
      sslcertfile: '/etc/ngcp-panel/api_ssl/api_ca.crt'
      sslcertkeyfile: '/etc/ngcp-panel/api_ssl/api_ca.key'
    serveradmin: support@sipwise.com
    servername: "\"myserver\""
    ssl_enable: yes
    sslcertfile: '/etc/ngcp-config/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
  frontend: no
  htpasswd:
    -
      pass: '{SHA}w4zj3mxbmynIQ1jsUEjSkN2z2pk='
      user: ngcpsoap
  logging:
    apache:
      acc:
        facility: daemon
        identity: oss
        level: info
      err:
        facility: local7
        level: info
    ossbss:
      facility: local0
      identity: provisioning
      level: DEBUG
    web:
      facility: local0
      level: DEBUG
  provisioning:
    allow_ip_as_domain: 1
    allow_numeric_usernames: 0
    auto_allow_cli: 1
    carrier:
      account_distribution_function: roundrobin
      prov_distribution_function: roundrobin
    credit_warnings:
    -
```

```

domain: example.com
recipients:
  - nobody@example.com
threshold: 1000
faxpw_min_char: 0
log_passwords: 0
no_logline_truncate: 0
pw_min_char: 6
routing:
  ac_regex: '[1-9]\d{0,4}'
  cc_regex: '[1-9]\d{0,3}'
  sn_regex: '[1-9]\d+'
tmpdir: '/tmp'

```

- `ossbss.frontend`: Enable/disable SOAP interface. Set value to `fcgi` to enable old SOAP interface.
- `ossbss.htpasswd`: Sets the username and SHA hashed password for SOAP access. You can generate the password using the following command: `htpasswd -nbs myuser mypassword`.
- `ossbss.provisioning.allow_ip_as_domain`: Allow or not allow IP address as SIP domain (0 is not allowed).
- `ossbss.provisioning.allow_numeric_usernames`: Allow or not allow numeric SIP username (0 is not allowed).
- `ossbss.provisioning.faxpw_min_char`: Minimum number of characters for fax passwords.
- `ossbss.provisioning.pw_min_char`: Minimum number of characters for sip passwords.
- `ossbss.provisioning.log_password`: Enable logging of passwords.
- `ossbss.provisioning.routing`: Regexp for allowed AC (Area Code), CC (Country Code) and SN (Subscriber Number).

### B.1.20 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.1.21 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.1.22 pushd

The following is the pushd section:

```
pushd:
  apns:
    enable: yes
    endpoint: api.push.apple.com
    endpoint_port: 0
    extra_instances:
      - certificate: '/etc/ngcp-config/ssl/PushCallkitCert.pem'
        enable: yes
        key: '/etc/ngcp-config/ssl/PushCallkitKey.pem'
        type: callkit
  http2_jwt:
    ec_key: '/etc/ngcp-config/ssl/AuthKey_ABCDE12345.pem'
    ec_key_id: 'ABCDE12345'
    enable: yes
    issuer: 'VWXYZ67890'
    tls_certificate: ''
    tls_key: ''
    topic: 'com.example.appID'
  legacy:
    certificate: '/etc/ngcp-config/ssl/PushChatCert.pem'
    feedback_endpoint: feedback.push.apple.com
    feedback_interval: '3600'
    key: '/etc/ngcp-config/ssl/PushChatKey.pem'
    socket_timeout: 0
  domains:
    - apns:
        endpoint: api.push.apple.com
        extra_instances:
```



```

- certificate: '/etc/ngcp-config/ssl/PushCallkitCert-example.com.pem'
  enable: no
  key: '/etc/ngcp-config/ssl/PushCallkitKey-example.com.pem'
  type: callkit
http2_jwt:
  ec_key: '/etc/ngcp-config/ssl/AuthKey_54321EDCBA.pem'
  ec_key_id: '54321EDCBA'
  issuer: '09876ZYXWV'
  tls_certificate: ''
  tls_key: ''
  topic: 'com.example.otherAppID'
legacy:
  certificate: '/etc/ngcp-config/ssl/PushChatCert-example.com.pem'
  feedback_endpoint: feedback.push.apple.com
  key: '/etc/ngcp-config/ssl/PushChatKey-example.com.pem'
domain: example.com
enable: yes
android:
  key: 'google_api_key_for_example.com_here'
enable: yes
android:
  enable: yes
  key: 'google_api_key_here'
priority:
  call: high
  groupchat: normal
  invite: normal
  message: normal
muc:
  exclude: []
  force_persistent: 'true'
  owner_on_join: 'true'
one_device_per_subscriber: no
port: 45060
processes: 4
ssl: yes
sslcertfile: /etc/ngcp-config/ssl/CAsigned.crt
sslcertkeyfile: /etc/ngcp-config/ssl/CAsigned.key
unique_device_ids: no

```

- `pushd.enable`: Enable/Disable the Push Notification feature.
- `pushd.apns.enable`: Enable/Disable Apple push notification.
- `pushd.apns.endpoint`: API endpoint hostname or address. Should be one of `api.push.apple.com` or `api.development.push.apple.com` for the newer HTTP2/JWT based protocol, or one of `gateway.push.apple.com` or `gateway.sandbox.push.apple.com` for the legacy protocol.

- `pushd.apns.endpoint_port`: API endpoint port. Normally 443 or alternatively 2197 for the newer HTTP2/JWT based protocol, or 2195 for the legacy protocol.
- `pushd.apns.legacy`: Contains all options specific to the legacy APNS protocol. Ignored when HTTP2/JWT is in use.
- `pushd.apns.legacy.certificate`: Specify the Apple certificate for push notification https requests from Sipwise C5 to an endpoint.
- `pushd.apns.legacy.key`: Specify the Apple key for push notification https requests from Sipwise C5 to an endpoint.
- `pushd.apns.legacy.feedback_endpoint`: Hostname or address of the APNS feedback service. Normally one of *feedback.push.apple.com* or *feedback.sandbox.push.apple.com*.
- `pushd.apns.legacy.feedback_interval`: How often to poll the feedback service, in seconds.
- `pushd.apns.extra_instances`: If the iOS app supports Callkit push notifications, they can be enabled here and the required separate certificate and key can be specified. Ignored if HTTP2/JWT is enabled.
- `pushd.http2_jwt`: Contains all options specific to the newer HTTP2/JWT based APNS API protocol.
- `pushd.http2_jwt.ec_key`: Name of file that contains the elliptic-curve (EC) cryptographic key provided by Apple, in PEM format.
- `pushd.http2_jwt.ec_key_id`: 10-digit identification string of the EC key in use.
- `pushd.http2_jwt.enable`: Master switch for the HTTP2/JWT based protocol. Disables the legacy protocol when enabled.
- `pushd.http2_jwt.issuer`: Issuer string for the JWT token. Normally the 10-digit team ID string for which the EC key was issued.
- `pushd.http2_jwt.tls_certificate`: Optional client certificate to use for the TLS connection.
- `pushd.http2_jwt.tls_key`: Optional private key for the client certificate to use for the TLS connection.
- `pushd.http2_jwt.topic`: Topic string for the JWT token. Normally the bundle ID for the iOS app.
- `pushd.android.enable`: Enable/Disable Google push notification.
- `pushd.android.key`: Specify the Google key for push notification https requests from Sipwise C5 to an endpoint.
- `pushd.domains`: Supports a separate set of push configurations (API keys, certificates, etc) for all subscribers of the given domain.
- `pushd.muc.exclude`: list of MUC room jids excluded from sending push notifications.
- `pushd.muc.force_persistent`: Enable/Disable MUC rooms to be persistent. Needed for Sipwise C5 app to work with other clients.
- `pushd.muc.owner_on_join`: Enable/Disable all MUC participants to be owners of the MUC room. Needed for Sipwise C5 app to work with other clients.
- `pushd.ssl`: The security protocol Sipwise C5 uses for https requests from the app in the push notification process.
- `pushd.sslcertfile`: The trusted certificate file purchased from a CA
- `pushd.sslcertkeyfile`: The key file that purchased from a CA
- `pushd.unique_device_ids`: Allows a subscriber to register the app and have the push notification enabled on more than one mobile device.

### B.1.23 qos

The QoS section allows configuring the ToS (Type of Service) feature:

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

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

---

#### Tip

The ToS byte includes both DSCP and ECN bits. So, specify the DSCP value multiplied by four ( $46 \times 4 = 184$ ) and, optionally, add the required ECN value to it (1, 2 or 3).

---

Set the `rtproxy.control_tos` parameter higher than zero to enable ToS.

### B.1.24 ngcp-rate-o-mat

The following is the *ngcp-rate-o-mat* section:

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

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

### B.1.25 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.1.26 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.1.27 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.1.28 rtpproxy

The following is the rtp proxy section:

```
rtpproxy:
  allow_userspace_only: yes
  cdr_logging_facility: ''
  control_tos: 0
  delete_delay: 30
  dtls_passive: no
  enable: yes
  final_timeout: 0
  firewall_iptables_chain: ''
  graphite:
    interval: 600
    prefix: rtpengine.
    server: ''
  log_level: '6'
  maxport: '40000'
  minport: '30000'
  num_threads: 0
  prefer_bind_on_internal: no
  recording:
    enable: no
    mp3_bitrate: '48000'
    log_level: '6'
    nfs_host: 192.168.1.1
    nfs_remote_path: /var/recordings
    output_dir: /var/lib/rtpengine-recording
    output_format: wav
    output_mixed: yes
    output_single: yes
    resample: no
    resample_to: '16000'
    spool_dir: /var/spool/rtpengine
  rtcp_logging_facility: ''
  rtp_timeout: '60'
  rtp_timeout_onhold: '3600'
```

- `rtpproxy.allow_userspace_only`: Enable/Disable the user space failover for rtpengine (*yes* means enable). By default rtpengine

works in kernel space.

- `rtpproxy.cdr_logging_facility`: If set, `rtengine` will produce a CDR-like syslog line after each call finishes. Must be set to a valid syslog facility string (such as `daemon` or `local0`).
- `rtpproxy.control_tos`: If higher than 0, the control messages port uses the configured ToS (Type of Service) bits. See the QoS section below for details.
- `rtpproxy.delete_delay`: After a call finishes, `rtengine` will wait this many seconds before cleaning up resources. Useful for possible late branched calls.
- `rtpproxy.dtls_passive`: If enabled, `rtengine` will always advertise itself as a passive role in DTLS setup. Useful in WebRTC scenarios if used behind NAT.
- `rtpproxy.final_timeout`: If set, any calls lasting longer than this many seconds will be terminated, no matter the circumstances.
- `rtpproxy.firewall_iptables_chain`: If set, `rtengine` will create an iptables rule for each individual media port opened in this chain.
- `rtpproxy.graphite.interval`: Interval in seconds between sending updates to the Graphite server.
- `rtpproxy.graphite.prefix`: Graphite keys will be prefixed with this string. Must include a separator character (such as a trailing dot) if one should be used.
- `rtpproxy.graphite.server`: Graphite server to send periodic statistics updates to. Disabled if set to an empty string. Must be in format `IP:port` or `hostname:port`.
- `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 `rtengine` for RTP traffic.
- `rtpproxy.minport`: Minimum port used by `rtengine` for RTP traffic.
- `rtpproxy.num_threads`: Number of worker threads to use. If set to 0, the number of CPU cores will be used.
- `rtpproxy.recording.enable`: Enable support for call recording.
- `rtpproxy.recording.mp3_bitrate`: If saving audio as MP3, bitrate of the output file.
- `rtpproxy.recording.log_level`: Same as `log_level` above, but for the recording daemon.
- `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.rtcp_logging_facility`: If set, `rtpproxy` will write the contents of all received RTCP packets to `syslog`. Must be set to a valid `syslog` facility string (such as `daemon` or `local0`).
- `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.1.29 security

The following is the security section. Usage of the firewall subsection is described in Section [17.2](#):

```
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 Sipwise C5 firewall (default: `yes`)
- `security.firewall.logging.file`: File firewall log messages go to (default: `/var/log/firewall.log`)
- `security.firewall.logging.tag`: String prepended to all log messages (internally `DROP` is added to any tag indicating the action triggering the message, default: `NGCPFW`)
- `security.firewall.nat_rules4`: Optional list of IPv4 firewall rules added to table `nat` using `iptables-persistent` syntax (default: `undef`)
- `security.firewall.nat_rules6`: Optional list of IPv6 firewall rules added to table `nat` using `iptables-persistent` syntax (default: `undef`)

- `security.firewall.policies.forward`: Default policy for iptables `FORWARD` chain (default: `DROP`)
- `security.firewall.policies.input`: Default policy for iptables `INPUT` chain (default: `DROP`)
- `security.firewall.policies.output`: Default policy for iptables `OUTPUT` chain (default: `ACCEPT`)
- `security.firewall.rules4`: Optional list of IPv4 firewall rules added to table `filter` using iptables-persistent syntax (default: `undef`)
- `security.firewall.rules6`: Optional list of IPv6 firewall rules added to table `filter` using iptables-persistent syntax (default: `undef`)

### B.1.30 `sems`

The following is the SEMS section:

```
sems:
  bindport: 5080
  conference:
    enable: yes
    max_participants: 10
  debug: no
  highport: 50000
  lowport: 40001
  media_processor_threads: 10
  prepaid:
    enable: yes
  sbc:
    calltimer_enable: yes
    calltimer_max: 3600
    outbound_timeout: 6000
  profile:
    - custom_header: []
      name: ngcp
    - custom_header: []
      name: ngcp_cf
  sdp_filter:
    codecs: PCMA,PCMU,telephone-event
    enable: yes
    mode: whitelist
  session_timer:
    enable: yes
    max_timer: 7200
    min_timer: 90
    session_expires: 300
  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`: Set the default maximum call duration. Note that this value can be overwritten in subscriber/customer/domain preferences setting `max_call_duration` parameter. Attention: in case of call transfer done by the callee, with `max_call_duration` set, the timer will be restarted from 0 for the new transferred call.
- `sems.sbc.outbound_timeout`: Set INVITE transaction timeout if the destination is not responding with provisional response message.
- `sems.sbc.profile.name`: Profile's name where to add the custom headers in `header_list` config parameter. Supported values: `ngcp` and `ngcp_cf`.
- `sems.sbc.profile.custom_header`: List of the custom headers that has to be whitelisted (default) by sems sbc in the corresponding profile.
- `sems.sbc.session_timer.enable`: If set to "no" all session timer headers are stripped off without considering the session timer related configuration done via the web interface. If set to "yes" the system uses the subscriber/peer configurations values set on the web interface. If set to "transparent" no validation is performed on Session Timer headers, they are ignored by SEMS and therefore negotiated end-to-end.
- `sems.vsc.*`: Define here the VSC codes.

### B.1.31 sms

This section provides configuration of **Short Message Service** on the NGCP. Description of the SMS module is provided earlier in this handbook [here](#).

In the below example you can see the default values of the configuration parameters.

```
sms:
  core:
    admin_port: '13000'
    smsbox_port: '13001'
  enable: no
  loglevel: '0'
  sendsms:
    max_parts_per_message: '5'
    port: '13002'
  smsc:
    dest_addr_npi: '1'
    dest_addr_ton: '1'
    enquire_link_interval: '58'
    host: 1.2.3.4
    id: default_smsc
    max_pending_submits: '10'
    no_dlr: yes
    password: password
    port: '2775'
    source_addr_npi: '1'
    source_addr_ton: '1'
    system_type: ''
    throughput: '5'
    transceiver_mode: '1'
    username: username
```

- `sms.core.admin_port`: Port number of admin interface of SMS core module (running on LB nodes).
- `sms.core.smsbox_port`: Port number used for internal communication between *bearerbox* module on LB nodes and *smsbox* module on PRX nodes. This is a listening port of the *bearerbox* module (running on LB nodes).
- `sms.enable`: Set to `yes` if you want to enable SMS module.
- `sms.loglevel`: Log level of SMS module; the default `0` will result in writing only the most important information into the log file.
- `sms.sendsms.max_parts_per_message`: If the SM needs to be sent as concatenated SM, this parameter sets the max. number of parts for a single (logical) message.
- `sms.sendsms.port`: Port number of *smsbox* module (running on PRX nodes).
- `sms.smsc`: Parameters of the connection to an SMSC

- `dest_addr_npi`: Telephony numbering plan indicator for the SM destination, as defined by standards (e.g. `1` stands for E.164)
- `dest_addr_ton`: Type of number for the SM destination, as defined by standards (e.g. `1` stands for "international" format)
- `enquire_link_interval`: Interval of SMSC link status check in seconds
- `host`: IP address of the SMSC
- `id`: An arbitrary string for identification of the SMSC; may be used in log files and for routing SMs.
- `max_pending_submits`: The maximum number of outstanding (i.e. not acknowledged) SMPP operations between Sipwise C5 and SMSC. As a guideline it is recommended that no more than 10 (default) SMPP messages are outstanding at any time.
- `no_dlr`: Do not request delivery report; when sending an SM and this parameter is set to `yes`, Sipwise C5 will not request DR for the message(s). May be required for some particular SMSCs, in order to avoid "Incorrect status report request parameter usage" error messages from the SMSC.
- `password`: This is the password used for authentication on the SMSC.
- `port`: Port number of the SMSC where Sipwise C5 will connect to.
- `source_addr_npi`: Telephony numbering plan indicator for the SM source, as defined by standards (e.g. `1` stands for E.164)
- `source_addr_ton`: Type of number for the SM source, as defined by standards (e.g. `1` stands for "international" format)
- `system_type`: Defines the SMSC client category in which Sipwise C5 belongs to; defaults to "VMA" (Voice Mail Alert) when no value is given. (No need to set any value)
- `throughput`: The max. number of messages per second that Sipwise C5 will send towards the SMSC. (Value type: float)
- `transceiver_mode`: If set to `1` (yes / true), Sipwise C5 will attempt to use a TRANSCEIVER mode connection to the SMSC. It uses the standard transmit port of the SMSC for receiving SMs too.
- `username`: This is the username used for authentication on the SMSC.

### B.1.32 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.1.33 sudo

The following is in the sudo section:

```
sudo:
  logging: no
  max_log_sessions: 0
```

- `logging`: enable/disable the I/O logging feature of sudo. See man page of `sudoreplay(8)`.
- `max_log_sessions`: when I/O logging is enabled, specifies how many log sessions per individual user sudo should keep before it starts overwriting old ones. The default `0` means no limit.

### B.1.34 telegraf

The following is in the telegraf section:

```
telegraf:
  interval: ~
```

- `telegraf.interval`: The number of seconds between each data gathering iteration, when the value is undefined, the code will fallback to use `monitoring.interval`.

### B.1.35 ngcp-witnessd

The following is the ngcp-witnessd tool section:

```
witnessd:
  debug: no
  interval: ~
  gather:
    asr_ner_statistics: yes
    kamilio_concurrent_calls: yes
    kamilio_dialog_active: yes
    kamilio_dialog_early: yes
    kamilio_dialog_incoming: yes
    kamilio_dialog_local: yes
    kamilio_dialog_outgoing: yes
    kamilio_dialog_relay: yes
    kamilio_shmem: yes
    kamilio_usrloc_regdevices: yes
    kamilio_usrloc_regusers: yes
    peering_groups: yes
    mpt_status: no
    mta_queue_len: yes
```

```
mysql_global_status: yes
mysql_slave_status: yes
mysql_replicate_check_interval: '3600'
mysql_replicate_check_tables:
- accounting
- billing
- carrier
- kamailio
- ngcp
- provisioning
- prosody
- rtcengine
- stats
mysql_replicate_ignore_tables:
- accounting.acc_backup
- accounting.acc_trash
- kamailio.acc_backup
- kamailio.acc_trash
- ngcp.pt_checksums_sp1
- ngcp.pt_checksums_sp2
- ngcp.pt_checksums
oss_provisioned_subscribers: yes
sip_responsiveness: yes
sip_stats_num_packets: yes
sip_stats_num_packets_perday: yes
sip_stats_partition_size: yes
```

- `witnessd.interval`: The number of seconds between each data gathering iteration, when the value is undefined, the code will fallback to use `monitoring.interval`.
- `witnessd.gather.asr_ner_statistics`: Enable ASR/NER statistics data.
- `witnessd.gather.kamailio_*`: Enable Kamailio statistics data.
- `witnessd.gather.mpt_status`: Enable MPT RAID status data.
- `witnessd.gather.mta_queue_len`: Enable MTA (exim4) queue length data.
- `witnessd.gather.mysql_global_status`: Enable global MySQL data.
- `witnessd.gather.mysql_slave_status`: Enable salave (replication) MySQL data.
- `witnessd.gather.mysql_replicate_check_interval`: MySQL replication check interval in seconds.
- `witnessd.gather.mysql_replicate_check_tables`: List of tables that need to be checked for replication issues.
- `witnessd.gather.mysql_replicate_ignore_tables`: List of tables that need to be ignored during replication check.
- `witnessd.gather.oss_provisioned_subscribers`: Enable OSS provisioned subscribers count data.
- `witnessd.gather.sip_*`: Enable SIP statistics data.

### B.1.36 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:
    enable: 1
  default_admin_settings:
    call_data: 0
    is_active: 1
    is_master: 0
    read_only: 0
    show_passwords: 1
  domain:
    preference_features: 1
    rewrite_features: 1
    vsc_features: 0
  fastcgi_workers: 2
  fax_features: 1
  fees_csv:
    element_order:
      - source
      - destination
      - direction
      - zone
      - zone_detail
      - onpeak_init_rate
      - onpeak_init_interval
      - onpeak_follow_rate
      - onpeak_follow_interval
      - offpeak_init_rate
      - offpeak_init_interval
      - offpeak_follow_rate
      - offpeak_follow_interval
      - use_free_time
  http_admin:
    autoprov_port: 1444
```

```
port: 1443
serveradmin: support@sipwise.com
servername: "\"myserver\""
ssl_enable: yes
sslcertfile: '/etc/ngcp-config/ssl/myserver.crt'
sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
http_csc:
  autoprov_bootstrap_port: 1445
  autoprov_port: 1444
  port: 443
  serveradmin: support@sipwise.com
  servername: "\"myserver\""
  ssl_enable: yes
  sslcertfile: '/etc/ngcp-config/ssl/myserver.crt'
  sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
logging:
  apache:
    acc:
      facility: daemon
      identity: oss
      level: info
    err:
      facility: local7
      level: info
peer:
  preference_features: 1
peering_features: 1
security:
  password_allow_recovery: 0
  password_max_length: 40
  password_min_length: 6
  password_musthave_digit: 0
  password_musthave_lowercase: 1
  password_musthave_specialchar: 0
  password_musthave_uppercase: 0
  password_sip_autogenerate: 0
  password_sip_expose_subadmin: 1
  password_web_autogenerate: 0
  password_web_expose_subadmin: 1
speed_dial_vsc_presets:
  vsc:
    - '*0'
    - '*1'
    - '*2'
    - '*3'
    - '*4'
    - '*5'
    - '*6'
```

```
- '*7'  
- '*8'  
- '*9'  
subscriber:  
  auto_allow_cli: 0  
  extension_features: 0  
  voicemail_features: 1
```

- `www_admin.http_admin.*`: Define the Administration interface and certificates.
- `www_admin.http_csc.*`: Define the Customers interface and certificates.
- `www_admin.contactmail`: Email to show in the GUI's Error page.

## B.2 constants.yml Overview

`/etc/ngcp-config/constants.yml` is one of the main configuration files that contains important (static) configuration parameters, like Sipwise C5 system-user data.



### Caution

Sipwise C5 platform administrator should not change content of `constants.yml` file unless absolutely necessary. Please contact Sipwise Support before changing any of the parameters within the `constants.yml` file!

---

## 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 Sipwise C5 system.

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

### Sample host configuration for Sipwise C5

A CE would look like:

```
self:  
  dbnode: '1'  
  eth0:  
    ip: 10.0.2.15  
    netmask: 255.255.255.0  
  type:  
    - web_ext  
    - web_int
```



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

## C MariaDB encryption

### C.1 Overview

MariaDB encryption support (officially called as "Data-at-Rest") enables innodb files, tables and binlogs data encryption so that if copied over the data is not usable without the master key. All the data accessed or modified by clients is encrypted/decrypted on the fly and transparent for the users. The feature comes with a price of 3% to 5% MariaDB performance loss (depending on the hardware, and CPU in particular).

### C.2 Configuration

There are new options in `constants.yml`

```
mysql:
  encryption:
    enable: yes
    encrypt_binlog: yes
    key: 1;a356c82422a9031f2e472047ad8220eeeea257d611849fbdc9f75b49933f75241
    threads: 1
```

**NOTE:** all changes in the configuration section will cause the MariaDB server to restart when `ngcpcfg` templates are applied.

- `mysql.encryption.enable`: Switch encryption on/off. Values: *yes,no*, Default: *yes*. When enabled, all tables are being encrypted, it takes from a few seconds to several minutes for MariaDB to encrypt all the data (depending on the overall size) and the encryption procedure is performed in the background, while all the data continues to be fully accessible. Also all new tables are created encrypted by default and it is not possible to disable encryption for specific tables as the encryption is *forced*.
- `mysql.encryption.encrypt_binlog`: Encrypt binlogs. Values: *yes,no*, Default: *yes*. While it is preferred to have this option enabled by default, for scenarios where binlog files need to be parsed, this option can be turned off. It is also possible to use `mysqlbinlog` with `--read-from-remote-server` option to read encrypted binlogs.
- `mysql.encryption.key`: Encryption key. The value is randomly generated during the `cfg-schema` upgrade when the option is added into `constants.yml`. The key is located in `/etc/mysql/keyfile` and normally **MUST NOT** be changed. Changing or losing the key permanently will render all the MariaDB tablespaces data (databases/tables) unusable.
- `mysql.encryption.threads`: Amount of encryption threads. Default: 1 How many MariaDB encryption threads should be running, this value depends on how many tables are created/removed or the encryption keys are rotated.

### C.3 What is not encrypted

- `slow-queries` log
- `mysqld.err` log

- `general queries log`, if enabled

#### C.4 Data restoration remarks

- When restoring data from an sql backup from another platform it is safe to do that as the currently used *encryption\_key* (inside `my.cnf`) is not affected this way.
- When copying `constants.yml` file from another platform and the encryption is enabled, the current *mysql.encryption.key* (inside `constants.yml`) must be restored in `constants.yml` to the same one the MariaDB server is originally started with or it will fail to start otherwise after `ngcpcfg apply`.

## D NGCP Internals

This chapter documents internals of Sipwise C5 that should not be usually needed, but might be helpful to understand the overall system.

### D.1 Pending reboot marker

The Sipwise C5 has the ability to mark a pending reboot for any server, using the file `/run/reboot-required`. As soon as the file exists, several components will report about a pending reboot to the end-user. The following components report about a pending reboot right now: `ngcp-status`, `ngcpcfg status`, `motd`, `ngcp-upgrade`. Also, `ngcp-upgrade` will NOT allow proceeding with an upgrade if it notices a pending reboot. It might affect `rtengine` dkms module building if there is a pending reboot requested by a newly installed kernel, etc.

### D.2 Redis id constants

The list of current Sipwise C5 Redis DB IDs:

| Service                    | central (role db) | local | Release          | Ticket   | Description                             |
|----------------------------|-------------------|-------|------------------|----------|---|
| <b>sems</b>                | -                 | 0     | mr3.7.1+         | -        | HA switchover                           |
| <b>rtengine</b>            | -                 | 1     | mr3.7.1+         | -        | HA switchover                           |
| <b>proxy</b>               | 2                 | -     | mr3.7.1+         | -        | Counter of hunting groups               |
| <b>proxy</b>               | 3                 | -     | mr3.7.1+         | -        | Concurrent dialog counters              |
| <b>proxy</b>               | -                 | 4     | mr3.7.1+         | -        | List of keys of the central counters    |
| <b>prosody</b>             | 5                 | -     | mr3.7.1+         | -        | XMPP cluster                            |
| <b>sems PBX</b>            | -                 | 6     | mr3.7.1+         | -        | HA switchover                           |
| <b>sems</b>                | 7                 | -     | mr4.1.1+         | MT#12707 | Sems malicious_call app                 |
| <b>captagent</b>           | -                 | 8     | mr4.1.1+ - mr7.1 | MT#15427 | Old captagent internal data (unused)    |
| <b>monitoring</b>          | 9                 | -     | mr4.3+ - mr5.5   | MT#31    | Old SNMP agent monitoring data (unused) |
| <b>proxy</b>               | 10                | -     | mr4.3+           | MT#16079 | SIP Loop detection                      |
| <b>ngcp-panel sessions</b> | -                 | 19    | mr6.3+           | TT#35523 | Panel login sessions                    |
| <b>proxy usrloc</b>        | 20                | -     | mr6.2+           | TT#32971 | SIP registrations                       |

| Service                 | central (role db) | local | Release | Ticket   | Description                   |
|-------------------------|-------------------|-------|---------|----------|-------------------------------|
| proxy acc               | -                 | 21    | mr6.2+  | TT#32971 | Accounting records            |
| proxy auth              | -                 | 22    | mr6.2+  | TT#32971 | Subscriber data               |
| proxy dialog            | -                 | 23    | mr6.2+  | TT#34100 | Dialog data                   |
| lb topos                | -                 | 24    | mr6.5+  | TT#40617 | Topos data                    |
| proxy pbx               | 25                | -     | mr8.0+  | TT#64404 | PBX in use by each subscriber |
| proxy 181 HIH           | 26                | -     | mr8.5+  | TT#89301 | HIH stored for 181 messages   |
| websocket               | -                 | 30    | mr7.1+  | TT#49703 | Internal data                 |
| websocket monitors      | -                 | 31    | mr7.1+  | TT#49703 | Monitors                      |
| websocket subscriptions | -                 | 32    | mr7.1+  | TT#49703 | Subscriptions                 |

### D.3 InfluxDB monitoring keys

The *InfluxDB ngcp* monitoring database contains time series of several monitoring sources. The following are some of the current measurements:

|            |   |
|------------|---|
| node       | Cluster node information.               |
| memory     | System memory information.              |
| proc_count | Process counts.                         |
| monit      | Monit supervised processes information. |
| mail       | MTA information.                        |
| mysql      | MySQL database information.             |
| kamailio   | Kamailio statistics information.        |
| sip        | SIP statistics information.             |

The *node* measurement contains the following fields:

|               |  |
|---------------|--|
| active        | Cluster node HA state (boolean: 1/0).                          |
| hb_proc_state | Cluster node GCS/CRM 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).                  |

The *monit* measurement contains the following fields:

|              |                     |
|--------------|---------------------|
| 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* measurement contains the following fields:

|                            |                                      |
|----------------------------|--------------------------------------|
| last_io_error              | Last IO error description.           |
| last_sql_error             | Last SQL error description.          |
| queries_per_second_average | Average of queries per second.       |
| replication_discrepancies  | Number of replication discrepancies. |

## D.4 Preferences

### D.4.1 Tables

Currently available tables for preferences are

`provisioning.voip_preferences`: contains all available preferences, do not contain user data. `provisioning.voip_preferences_group_names`: contains preference group names, so the preferences can be put into groups. `provisioning.voip_preferences_enum`: contains enum values for preferences, do not contain user data.

The following tables contain user data and depend on `voip_preferences` and optionally on `voip_preferences_enum`:

`provisioning.voip_dev_preferences`: PBX device model preferences `provisioning.voip_devprof_preferences`: PBX device profile preferences `provisioning.voip_dom_preferences`: domain preferences, replicated by triggers to `kamailio.dom_preferences` `provisioning.voip_contract_preferences`: customer preferences, replicated by triggers to `kamailio.contract_preferences` `provisioning.voip_peer_preferences`: peering server preferences, replicated by triggers to `kamailio.peer_preferences` `provisioning.voip_prof_preferences`: subscriber profile preferences `provisioning.voip_reseller_preferences`: reseller preferences `provisioning.voip_usr_preferences`: subscriber preferences, replicated by triggers to `kamailio.usr_preferences`

### D.4.2 Columns

**voip\_preferences** id: primary key, used in user tables as the foreign key `voip_preference_groups_id`: preference group id attribute: preference name label: tooltip that can be used as a mouseover tooltip on the UI type: 0 - *string*, 1 - *integer/boolean* max\_occur: how many preferences with the name are allowed 0: list, 1: only one `usr_pref`: de-

fines if the preference can be used in subscribers `prof_pref`: defines if the preference can be used in subscriber profiles  
`dom_pref`: defines if the preference can be used in domains `peer_pref`: defines if the preference can be used in peer-  
 ing servers `contract_pref`: defines if the preference can be used in customers `contract_location_pref`: defines  
 if the preference can be used in customer locations `dev_pref`: defines if the preference can be used in PBX device models  
`devprof_pref`: defines if the preference can be used in PBX device profiles `fielddev_pref`: defines if the preference can  
 be used in PBX devices that are assigned to a subscriber `modify_timestamp`: preference last modification time `internal`:  
 preference if for internal use only and not shown in the UI/API `expose_to_customer`: unused, as now there are dedicated  
 customer preferences table `data_type`: data type *enum, boolean, int, string* `read_only`: ready only flag `description`:  
 long description of the preference `dynamic`: set to 1 if it is a custom preference that is created by a user (usually for a PBX  
 device model that requires specific preferences) but can be used for all preferences when needed `reseller_pref`: defines if  
 the preference can be used in resellers

### D.4.3 Enum

All tables are in database "provisioning".

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

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

voip\_preferences\_enum is:

```
id
  primary key
```

```
preference_id
  Reference to table voip_preferences.
```

```
label
  A label to be displayed in frontends.
```

```
value
  Value that will be written to voip_[usr|dom|peer]_preferences.value
  if it is NOT NULL. Will not be written if it IS NULL. This can be
  used to implement a "default value" for a preference that is visible
  in frontends as such (will be listed first if nothing is actually
  selected), but will not be written to
```

voip\_[usr|dom|peer]\_preferences.value. Usually forcing a domain or peer default. Should also be named 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*.



## E Kamilio pv\_headers module

This chapter documents the new kamilio "pv\_headers" modules introduced in Sipwise C5 starting from version mr7.0.1.

### E.1 Module overview

This new module enables storing all headers in XAVP to freely modify them in the kamilio logic and only apply them once when it's time for the packet to be routed outside. The main goal of the module is to offload the intermediate header processing into the XAVP dynamic container as well as provide with high level methods and pseudovariables to simplify SIP message header modifications.

In few words:

- as soon as a SIP message enters the proxy, kamilio reads all the headers (using the function "pv\_collect\_headers()") and stores them in an XAVP called "headers".
- starting from this point all the header changes are directly performed on the "headers" XAVP. For example the From header is available at `$xavp(headers[0]⇒From[0])`.
- right before the SIP message leaves the proxy, kamilio writes back all the headers changes (using the function "pv\_apply\_headers()").

RURI and the headers listed in the module parameter "skip\_headers" are left untouched and not saved in the XAVP. Therefore they should be handled in the usual way.

### E.2 Template changes

As described before in the upgrade procedures, the module is enabled by default in kamilio proxy and all the templates have been already updated to use this new logic. Before proceeding with the upgrade, it is essential that the customtt/patchtt files you have in place are updated to this new format.

Here just some few examples of what has been changed in the proxy templates:

- variables \$fu, \$fU, \$fd, \$fn, \$ft have been substituted by \$x\_fu, \$x\_fU, \$x\_fd, \$x\_fn, \$x\_ft
- variables \$tu, \$tU, \$td, \$tn, \$tt have been substituted by \$x\_tu, \$x\_tU, \$x\_td, \$x\_tn, \$x\_tt
- variables \$rr, \$rs have been substituted by \$x\_rr, \$x\_rs
- variables \$ua have been substituted by \$x\_hdr(User-Agent)
- variables \$ai have been substituted by \$x\_hdr(P-Asserted-Identity)
- variables \$pU, \$pd have been substituted by \$x\_hdr(P-Preferred-Identity)
- variables \$re have been substituted by \$x\_hdr(Remote-Party-ID)
- variables \$di have been substituted by \$x\_hdr(Diversion)

- variables `$ct` have been substituted by `$x_hdr(Contact)`
- `$hdr("name")` has been substituted by `$x_hdr("name")`
- `is_present_hf("name")` has been substituted by `$x_hdr(name)!= $null`
- `remove_hf("name")` has been substituted by `pv_remove_header("name")` function or `$(x_hdr(name)[*]) = $null`
- `append_hf("name: value\r\n")` has been substituted by `pv_append_header("name", "value") / pv_modify_header("name", "value")` functions or `$(x_hdr(name)[*]) = value`
- `t_check_status(code)` has been substituted by `$T_reply_code == code`
- `save("location")` has been updated in `save("location", "0x00", "$x_tu")`
- `sd_lookup("speed_dial")` has been updated in `sd_lookup("speed_dial", $x_fu)`
- added `pv_collect_headers()` and `pv_reset_headers()` functions in the dedicated `ROUTE_COLLECT_HDR` route
- added `pv_apply_headers()` function in the dedicated `ROUTE_APPLY_HDR` route
- added `pv_reset_headers()` function in the following routing sections

## E.3 Module documentation

### E.3.1 Parameters

#### **xavp\_name (string)**

Name of the XAVP where the collected headers are stored.

Default: headers

```
modparam("pv_headers", "xavp_name", "headers")
```

Result:

```
$xavp(headers[0]=>From)
$xavp(headers[0]=>To)
$xavp(headers[0]=>Call-ID)
....
```

#### **skip\_headers (string)**

A comma separated headers list that must be excluded from processing (they are skipped when `pv_apply_headers()` changes the sip message headers). If the parameter is not set then the "Default" list is used. If the parameter is set to an empty string then all the sip message headers are processed.

Default: Record-Route,Via,Route,Content-Length,Max-Forwards

#### **split\_headers (string)**

A comma separated headers list that must be split into multi headers if their value is a comma separated list. If the parameter is not set then the "Default" is used. If the parameter is set to an empty string then no headers are split.

Default: None

```
modparam("pv_headers", "split_headers", "Diversion")
```

Result:

Received Diversion header:

```
Diversion: <user1@test.local>,<user2@test.local>,<user3@test.local>
```

After split:

```
Diversion: <user1@test.local>
```

```
Diversion: <user2@test.local>
```

```
Diversion: <user3@test.local>
```

Becomes handy `if` used together with `pv_modify_header()` or `pv_remove_header()` to change or remove value 2 `for` instance.

## E.3.2 Functions

### `pv_collect_headers()`

This function collects all headers from the message into the XAVP. It should be used preferably just when the sip message is received by kamailio.

Returns:

- 1 - on success
- -1 - if there were errors

### `pv_apply_headers()`

This function applies the current XAVP headers state to the real headers and should be called only once per branch when the message is about to leave kamailio.

The following rules apply:

- all headers in the XAVP except for ones provided in the "skip\_headers" parameter and From/To are recreated in the sip message.
- From/To headers are processed by the uac module if it is loaded.
- From/To headers are not changed in the reply messages.
- headers with NULL value are removed if exist in the sip message.
- the initial order of the sip headers is preserved.

Usage:

```
if (pv_apply_headers ())
{
    "success"
}
else
{
    "errors"
}
```

### **pv\_reset\_headers()**

This function resets the current XAVP headers list and enables `pv_collect_headers()` and `pv_apply_headers()` to be called again in the same branch.

Usage:

```
if (pv_reset_headers ())
{
    "success"
}
else
{
    "errors"
}
```

### **pv\_check\_header(hname)**

This function checks if the header already exists in the XAVP. It can be freely called from anywhere, but only after `pv_collect_headers()`.

Usage:

```
if (pv_check_header (hname) )
{
    "exists"
}
else
{
    "does not exist"
}
```

### **pv\_append\_header(hname, hvalue)**

This function appends a new header into the XAVP. It can be freely called from anywhere, but only after `pv_collect_headers()`.

Please note that subsequent "pv\_append\_header" calls will result in multiple headers. If the provided "hvalue" is \$null then the header is added into the XAVP but it is not going to be added into the message.

Usage:

```
if (pv_append_header(hname, hvalue))
{
    "appended"
}
else
{
    "errors"
}
```

### **pv\_modify\_header(hname, hvalue)**

This function modifies an existing header in the XAVP. It can be freely called from anywhere, but only after pv\_collect\_headers(). Please note that if the header does not exist it will be explicitly appended. If there are multiple headers with the same name only the first one will be affected. If the provided header value is \$null then the header is modified in the XAVP then it is removed from the sip message when pv\_apply\_headers() is called.

Usage:

```
if (pv_modify_header(hname, hvalue))
{
    "modified"
}
else
{
    "errors"
}
```

### **pv\_modify\_header(hname, idx, hvalue)**

This function works similar to pv\_modify\_header(hname, hvalue) but should be used when there are multiple headers with the same name one of them to be modified. Index order is top to bottom.

Usage:

```
if (pv_modify_header(hname, idx, hvalue))
{
    "modified"
}
else
{
```

```

    "errors"
}

```

### pv\_remove\_header(hname)

This function removes an existing header from the XAVP. It can be freely called from anywhere, but only after pv\_collect\_headers(). If there are multiple headers with the same name all of them are removed. It returns -1 if the header does not exist.

Usage:

```

if (pv_remove_header(hname, hvalue))
{
    "removed"
}
else
{
    "does not exist or errors"
}

```

### pv\_remove\_header(hname, idx, hvalue)

This function works similar to pv\_remove\_header(hname, hvalue) but should be used when there are multiple headers with the same name one of them to be removed. Index order is top to bottom.

Usage:

```

if (pv_remove_header(hname, idx, hvalue))
{
    "removed"
}
else
{
    "does not exist or errors"
}

```

## E.3.3 Pseudovariabes

### \$x\_hdr

This pseudovariabes is used to append/modify/remove headers by their name and can be used instead of the pv\_append\_header(), pv\_modify\_header(), pv\_remove\_header() functions.

Usage:

- append header "X-Header" with value "example". NOTE: It always appends a header, even there is already one with the same

name

```
$x_hdr(X-Header) = "example";
```

- modify header "X-Header" with index 0. Returns an error if there is no such index

```
$(x_hdr(X-Header)[0]) = "example";
```

- remove all occurrences of header "X-Header" and append one with value "example"

```
$(x_hdr(X-Header)[*]) = "example";
```

- remove header "X-Header" with index 2 (if there are multiple headers). Returns an error if there is no such index

```
$(x_hdr(X-Header)[2]) = $null;
```

- remove all occurrences of the header. Does not produce an error if there is no such header

```
$(x_hdr(X-Header)[*]) = $null;
```

- retrieve a value of header "X-Header" with index 0, otherwise \$null

```
$var(test) = $x_hdr(X-Header);
```

- retrieve a value of header "X-Header" with index 0 otherwise \$null

```
$var(test) = $x_hdr(X-Header)[*];
```

- retrieve a value of header "X-Header" with index 2 otherwise \$null

```
$var(test) = $(x_hdr(X-Header)[2]);
```

### **\$x\_fu, \$x\_tu**

These pseudovariabes are used to modify/retrieve the "From" and "To" headers.

Usage:

- modify the header

```
$x_fu = "User1 <440001@example.local>";
```

- retrieve a value of the header

```
$var(test) = $x_fu;
```

- \$x\_tu usage is the same

### **\$x\_fU, \$x\_tU**

These pseudovariabes are used to modify/retrieve the username part of the "From" and "To" headers.

Usage:

- modify the username part

```
$x_fU = "440001";
```

- retrieve the username part

```
$var(test) = $x_fU;
```

- \$x\_tU usage is the same



**\$x\_fd, \$x\_td**

These pseudovariabes are used to modify/retrieve the domain part of the "From" and "To" headers.

Usage:

- modify the domain part

```
$x_fd = "example.local";
```

- retrieve the domain part

```
$var(test) = $x_fd;
```

- \$x\_td usage is the same

**\$x\_fn, \$x\_tn**

These pseudovariabes are used to modify/retrieve the display part of the "From" and "To" headers.

Usage:

- modify the username part

```
$x_fn = "User1";
```

- retrieve the domain part

```
$var(test) = $x_fn;
```

- \$x\_tn usage is the same

**\$x\_ft, \$x\_tt**

These pseudovariabes are used to retrieve the tag part of the "From" and "To" headers.

Usage:

- retrieve the tag part

```
$var(test) = $x_ft;
```

- \$x\_tt usage is the same

### **\$x\_rs, \$x\_rr**

These pseudovariabes are used to modify/retrieve or change "status" and "code" of the SIP reply NOTE: Only messages with reply status > 300 can be changed as well as reply status 1xx and 2xx cannot be set

Usage:

- modify the reply status

```
$x_rs = 486
```

- retrieve the reply status

```
$var(test) = $x_rs;
```

- modify the reply reason

```
$x_rr = "Custom Reason"
```

- retrieve the reply reason

```
$var(test) = $x_rr;
```

## F Extra Configuration Scenarios

### F.1 AudioCodes devices workaround

Old AudioCodes devices suffer from a problem where they replace 127.0.0.1 address in Record-Route headers (added by Sipwise C5's internal components) with the device's IP address. Supposedly, the whole range of AudioCodes devices with a firmware version below 6.8.X are affected. As a workaround, you may enable the topos feature to stop sending Record-Route headers out. To achieve this, execute the following commands:

```
ngcpcfg set /etc/ngcp-config/config.yml kamailio.lb.security.topos.enable=yes
ngcpcfg apply 'enable topos for audiocodes devs workaround'
```

### F.2 "Debug Proxy" for troubleshooting



#### Important

This functionality only makes sense on Sipwise C5 CARRIER appliance environment that has an inactive proxy node available.

---

In order to troubleshoot/debug/capture a scenario, the "debug proxy" allows defining a list of "debug subscribers" that will be matched against From/To headers for every SIP message on a specific lb node. If matched that SIP message will be delivered to the assigned proxy node. In summary, any call to/from that subscriber will be easily traced since no calls are delivered by default to an inactive "debug proxy" node. Also, you can enable extra debug levels on the "debug proxy" node which will NOT affect production traffic on the platform.



#### Important

The subscribers have to be specified in the same format as they are received on Kamailio LB (before all the rewrite rules applied).

---



#### Important

In the following examples both proxy node *prx99a* and *prx99b* must be set to **inactive** nodes *hosts.prx99a.status=inactive* and *hosts.prx99b.status=inactive* in *network.yml*!

---

To enable the feature for INACTIVE *prx99* proxy pair, execute:

```
ngcpcfg set /etc/ngcp-config/network.yml hosts.prx99a.status=inactive # for the safety
ngcpcfg set /etc/ngcp-config/network.yml hosts.prx99b.status=inactive # for the safety
```

```
ngcpcfg set /etc/ngcp-config/config.yml kamailio.lb.debug_uri.enable=yes
ngcpcfg apply 'Enable debug proxy on node prx99'
ngcpcfg push-parallel all
```

And make sure *prx99* active node has this new config applied.

There's a command-line tool available to manage the subscriber list. An example of use:

```
ngcp-debug-subscriber add lb01 +4310001000@example.org sipuser@example.org prx99
ngcp-debug-subscriber delete lb01 +4310001000@example.org
ngcp-debug-subscriber list lb01
```

Be aware that the list of debug subscribers belongs to just one lb pair, the info is kept in REDIS *local* database, it is necessary to survive LB restarts and/or HA switchovers. To skip saving in REDIS *local* database, specifying the option *--no-store* (in this case the information will stay in memory only and will be void on Kamailio LB restart):

```
ngcp-debug-subscriber add --no-store lb01 +10001042@example.org prx99
```

See more available options in general and per-action help messages:

```
ngcp-debug-subscriber --help
ngcp-debug-subscriber add --help
```



### Warning

It is recommended to keep the amount of "debug subscribers" as small as possible (for performance reasons).

---

The "debug subscribers" is kept in a kamailio htable. htable index size value can be changed if necessary in config.yml using the option *kamailio.lb.debug\_uri.htable\_idx\_size*. This is **not** the maximum size. From Kamailio htable documentation:

```
size - number to control how many slots (buckets) to create for the hash table.
A larger value means more slots with a higher probability for fewer collisions.
The actual number of slots (or buckets) created for the table is 2^size.
The possible range for this value is from 2 to 31, smaller or larger values
will be increased to 3 (8 slots) or decreased to 14 (16384 slots).
Note that each slot can store more than one item, when there are collisions of
hash ids computed for keys. The items in the same slot are stored in a linked list.
In other words, the size is not setting a limit of how many items can be
stored in a hash table, as long as there is enough free shared memory,
new items can be added.
```

The default value *kamailio.lb.debug\_uri.htable\_idx\_size=4* is enough for all the use cases in production.