FRAFOS ABC SBC Handbook

Release 4.6

FRAFOS GmbH

Jan 31, 2022

Contents

1 About the ABC Session Border Controller **1**
- 1.1 How to Start? . . . 2
- 1.2 Credits . . . 3

2 Release Notes **4**
- 2.1 Release Notes for ABC SBC version 4.6 . . . 4
- 2.2 Release Notes for ABC Monitor version 4.6 . . . 5

3 Introduction **7**
- 3.1 A Brief Introduction to History and Architecture of SIP . . . 7
- 3.2 What is a Session Border Controller (SBC)? . . . 10
  - 3.2.1 General Behavior of SBCs . . . 10
  - 3.2.2 General Deployment Scenarios of SBCs . . . 12
- 3.3 Do You Need an SBC? . . . 13
- 3.4 ABC SBC Networking Concepts . . . 14
  - 3.4.1 Network Topology . . . 14
  - 3.4.2 SBC Interfaces . . . 14
  - 3.4.3 Call Agents . . . 15
  - 3.4.4 Realms . . . 15
  - 3.4.5 A-B-C rules . . . 15

4 Practical Guide to the ABC SBC **19**
- 4.1 Network Planning Guidelines . . . 19
  - 4.1.1 Topology Model . . . 19
  - 4.1.2 SBC Logic . . . 21
  - 4.1.3 Security Policies . . . 23
  - 4.1.4 Capacity planning . . . 24
  - 4.1.5 IT Integration . . . 25
- 4.2 Planning Checklists . . . 26
- 4.3 A Typical SBC Configuration Example . . . 28
  - 4.3.1 Identifying Network topology . . . 28
  - 4.3.2 Describing ABC SBC Realms and Call Agents . . . 29
  - 4.3.3 Configuring Registration Cache and Throttling . . . 32
  - 4.3.4 SIP Routing . . . 34
  - 4.3.5 Configuring NAT Handling and Media Anchoring . . . 36
  - 4.3.6 Configuring transparent dialog IDs . . . 39
  - 4.3.7 Setting up tracing . . . 40
  - 4.3.8 Summary of rules . . . 41
  - 4.3.9 Setting Call Limits . . . 41
  - 4.3.10 Blacklisting specific IPs and User Agents . . . 43
  - 4.3.11 Handling P-Asserted-Identity . . . 44
  - 4.3.12 Where to go from here . . . 45

5 Installing the ABC SBC **46**
- 5.1 Types of Installations: Hardware-based, Container and Cloud-based . . . 46
- 5.2 Hardware Requirements . . . 46
- 5.3 OS Tuning . . . 47

5.3.1 Disk Partitioning . . . 47
5.4 Deployment Modes . . . 47
5.4.1 Single Node Mode . . . 48
5.4.2 High Available (HA) Pair Mode . . . 48
5.4.3 Cluster based solution . . . 48
5.5 Installation Procedure . . . 48
5.6 Installation Procedure - Standard ABC SBC install . . . 48
5.6.1 Repository Access and Configuration . . . 49
5.6.2 ABC SBC Package Installation . . . 50
5.7 Installation Procedure - Container ABC SBC install . . . 51
5.7.1 Unpack the container image . . . 52
5.7.2 Prepare dir for persistent data . . . 52
5.7.3 Create systemd service file . . . 52
5.7.4 Start and manage the container . . . 53
5.7.5 Connect to the container console . . . 53
5.8 Initial Configuration . . . 53
5.8.1 SBC Interfaces Overview . . . 53
5.8.2 Web GUI Configuration (Choosing the Cluster Config Master) . . . 54
5.9 Setting Up Web Interface Access and User Accounts . . . 55
5.9.1 Default User Accounts . . . 55
5.10 ABC SBC License . . . 56
5.11 Interface Configuration . . . 57
5.11.1 Physical and System Interfaces . . . 57
5.11.2 SBC Interfaces . . . 58
5.11.3 Retro Compatibility . . . 60
5.12 TLS profiles Configuration . . . 61
5.12.1 TLS profile options . . . 61
5.12.2 Certificate requirements . . . 62
5.12.3 Let's encrypt certbot . . . 62
5.13 Hardware Specific Configurations . . . 63
5.13.1 MariaDB Server Settings . . . 63
5.13.2 Network adapters . . . 64
5.13.3 Configuration of SBC Number of Threads . . . 64
5.14 Last ABC SBC Installation Steps . . . 65
5.15 ABC Monitor Installation (optional) . . . 66
5.15.1 ABC Monitor recommended server configuration . . . 66
5.15.2 Repository Access and Configuration . . . 67
5.15.3 ABC Monitor Package Installation . . . 67
5.16 ABC Monitor Container Installation (optional) . . . 68
5.16.1 ABC Monitor Initial Configuration . . . 68
5.17 ABC Monitor LDAP Installation (optional) . . . 70
5.18 ABC Monitor Installation Off AWS (optional) . . . 70
5.19 G729 Codec Installation (optional) . . . 71

6 General ABC Configuration Guide 73
6.1 Physical, System and SBC Interfaces . . . 73
6.2 Defining Rules . . . 74
6.2.1 Condition Types . . . 75
6.2.2 Condition Operators . . . 76
6.2.3 Condition Values and Regular Expressions . . . 78
6.2.4 Actions . . . 78
6.3 Using Replacements in Rules . . . 79
6.3.1 Example Use of Replacement Expressions . . . 82
6.4 Using Regular Expression Backreferences in Rules . . . 84
6.5 Binding Rules together with Call Variables . . . 85
6.6 SIP Routing . . . 86
6.6.1 Routing Rules (B) . . . 87
6.6.2 Static Routes . . . 88

6.6.3 Table-based Dynamic Routes . . . 90
6.6.4 Request-URI Based Routes . . . 92
6.6.5 Determination of the IP destination and Next-hop Load-Balancing . . . 93
6.6.6 IP Blacklisting: Adaptive Availability Management . . . 96
6.6.7 SIP Routing by Example . . . 98
6.7 SIP Mediation . . . 101
6.7.1 Why is SIP Mediation Needed? . . . 102
6.7.2 Request-URI Modifications . . . 103
6.7.3 Changing Identity . . . 104
6.7.4 SIP Header Processing . . . 105
6.7.5 Early Media, Ring Back Tone and Forking . . . 107
6.7.6 Call transfers . . . 109
6.7.7 INVITE with Replaces handling . . . 110
6.7.8 Mapping Dialog-IDs in INVITEs with Replaces . . . 110
6.7.9 Other mediation actions . . . 110
6.8 SDP Mediation . . . 111
6.8.1 Codec Signalling . . . 112
6.8.2 Media Type Filtering . . . 112
6.8.3 CODEC Filtering . . . 113
6.8.4 CODEC Preference . . . 114
6.8.5 SDP Bandwidth attribute limiting . . . 115
6.9 Media Handling . . . 116
6.9.1 Introduction . . . 116
6.9.2 Media Anchoring (RTP Relay) . . . 117
6.9.3 RTP and SRTP Interworking . . . 120
6.9.4 SRTP End to End encryption . . . 120
6.9.5 Transcoding . . . 120
6.9.6 Audio Recording . . . 121
6.9.7 Playing Audio Announcements . . . 123
6.9.8 Onboard Conferencing . . . 124
6.10 NAT Traversal . . . 126
6.10.1 NAT Traversal Configuration Example . . . 128
6.11 Registration Caching and Handling . . . 130
6.11.1 Registration Handling Configuration Options . . . 131
6.11.2 Registrar off-load . . . 134
6.11.3 Registration Caching and Handling by Example . . . 135
6.11.4 Registration Agent . . . 139
6.12 Call Data Records (CDRs) . . . 141
6.12.1 CDRs Location . . . 141
6.12.2 CDR Format . . . 141
6.12.3 Access to CDRs . . . 142
6.12.4 Customised CDR Records . . . 142
6.13 Advanced Use Cases with Provisioned Data . . . 144
6.13.1 RESTful Interface . . . 144
6.13.2 Provisioned Tables . . . 147
6.13.3 ENUM Queries . . . 158
6.14 SIP-WebRTC Gateway . . . 159
6.14.1 WebRTC Network Architecture and Protocols . . . 161
6.14.2 WebRTC Network Configuration . . . 163
6.14.3 WebRTC Credentials Configuration . . . 165
6.14.4 WebRTC Rules Configuration . . . 166
6.14.5 WebRTC Interoperability Recommendations . . . 170
6.15 Amazon Elastic Cloud Configuration Cookbook . . . 171
6.15.1 Before you Start: Prerequisites and Important Warnings . . . 171
6.15.2 Quick Start Using Cloud Formation . . . 172
6.15.3 Quick Start: Launch Single Instance . . . 173
6.15.4 Updating License . . . 173
6.15.5 Introducing Geographic Dispersion . . . 174

6.15.6 Monitoring the Autoscaling Cluster Using CloudWatch . . . 177
6.15.7 Performance Recommendations . . . 180
6.16 Template parameters . . . 180
6.16.1 Definition of Template Parameter . . . 180
6.16.2 Set specific values for Template Parameters . . . 181

7 ABC SBC System administration 183
7.1 User Management . . . 183
7.1.1 GUI User Management . . . 183
7.1.2 CLI User Management . . . 184
7.2 Server Administration . . . 185
7.3 Backup and Restore Operations . . . 185
7.3.1 ABC SBC Configuration Management . . . 185
7.3.2 ABC SBC Configuration Backup . . . 186
7.3.3 ABC SBC Recovery Procedure . . . 187
7.3.4 Manual Backup of the Complete SBC Configuration . . . 188
7.3.5 Manual Restore of the Complete SBC Configuration . . . 189
7.4 ABC Monitor Backup and Restore Operations . . . 191
7.4.1 ABC Monitor Configuration Backup . . . 191
7.4.2 ABC Monitor Configuration Restore . . . 191
7.5 Howto setup a Semi-redundant CCM on ABC SBC . . . 191
7.5.1 Setup primary CCM node . . . 192
7.5.2 Setup backup CCM node . . . 192
7.5.3 Configure configuration snapshot backups . . . 192
7.5.4 Setup configuration backups transfer to backup CCM node . . . 193
7.5.5 Steps to make the backup CCM available in case of primary CCM node failure . . . 194
7.5.6 Steps to be done on SBC nodes to start using new CCM . . . 194
7.5.7 Additional steps and checks . . . 194
7.6 Upgrade Procedure . . . 195
7.6.1 Standard ABC SBC upgrade . . . 195
7.6.2 Container ABC SBC upgrade . . . 195
7.7 ABC Monitor Upgrade Procedure . . . 196
7.8 SBC Dimensioning and Performance Tuning . . . 197
7.8.1 Trunking Use Case . . . 198
7.8.2 Trunking with Transcoding . . . 198
7.8.3 Traffic Estimates for Residential VoIP . . . 198
7.8.4 Performance Tuning . . . 199

8 Monitoring and Troubleshooting 200
8.1 Overview of Monitoring and Troubleshooting Techniques . . . 200
8.2 ABC Monitor (Optional) . . . 201
8.2.1 Events (optional) . . . 203
8.2.2 HOWTO Find a Needle in the Haystack: Iterative Event Filtering . . . 211
8.2.3 Using Filters . . . 215
8.2.4 Overview Dashboard . . . 217
8.2.5 Calls Dashboard . . . 218
8.2.6 Registration Dashboard . . . 221
8.2.7 Connectivity CA Dashboard . . . 222
8.2.8 Security Dashboard . . . 225
8.2.9 Exceeded Limits Dashboards . . . 227
8.2.10 System Dashboard . . . 230
8.2.11 Network and Statistics Dashboard . . . 231
8.2.12 Diagnostics Dashboard . . . 232
8.2.13 Monitor Troubleshooting . . . 233
8.3 Live ABC SBC Information . . . 234
8.3.1 Registration Cache . . . 234
8.3.2 Live Calls . . . 235
8.3.3 Destination Blacklists . . . 235

8.3.4 User Recent Traffic . . . 236
8.4 Using SNMP for Measurements and Monitoring . . . 237
8.4.1 General Statistics . . . 237
8.4.2 Statistics per Realm / Call Agent . . . 238
8.4.3 Call Agent destination status . . . 238
8.4.4 Interfaces statistic . . . 239
8.4.5 User Defined Counters . . . 239
8.4.6 SNMP traps . . . 240
8.5 Command-line SBC Process Management . . . 240
8.5.1 Process Management using Systemd . . . 240
8.5.2 SEMS – the SIP and RTP processing Daemon . . . 241
8.5.3 REDIS – the Real-time Database . . . 241
8.6 Additional Sources of Diagnostics Information . . . 242

9 Securing SIP Networks using ABC SBC and ABC Monitor (optional) 243
9.1 SIP Security Principles: Collect, Analyze and Police . . . 243
9.2 Police: Devising Security Rules in the ABC SBC . . . 245
9.2.1 Manual IP-layer Blocking . . . 247
9.2.2 Automatic IP Address Blocking . . . 248
9.2.3 Automatic Proactive Blocking: Greylisting . . . 252
9.2.4 Manual SIP Traffic Blocking . . . 255
9.2.5 Traffic Limiting and Shaping . . . 260
9.2.6 Call Duration Control . . . 264
9.3 Collect Events: Gathering Usage Data in the ABC Monitor . . . 265
9.3.1 Reporting Security Events . . . 266
9.3.2 Setting up Diagnostic Events . . . 266
9.4 Analyze: Finding Patterns in Events using the ABC Monitor . . . 267
9.4.1 Password Guessing Attacks . . . 268
9.4.2 Scanning Attacks . . . 269
9.4.3 Denial-of-Service Attacks . . . 270
9.4.4 Distributed Attacks . . . 272
9.4.5 Dial-out Attempts . . . 272
9.5 Practices for Devising Secure Rule-basis . . . 273
9.5.1 Topology Hiding . . . 274
9.5.2 Devising a secure rule-base . . . 276

10 Preview of Experimental Features 280
10.1 Using Two-Factor Authentication for Users . . . 280
10.1.1 Prerequisites . . . 280
10.1.2 Rules for Two Factor Authentication Processing . . . 286
10.1.3 Rules for determining User Status and discriminating by it . . . 287
10.1.4 Routing Rule to Connect Two Factor Authentication Processing and User Discrimination 288
10.1.5 Scenario Modifications . . . 288
10.2 AWS: Reputation Lists . . . 288
10.2.1 Setting Up ABC SBC for Use of Reputation List on AWS . . . 289
10.2.2 Setting Up ABC Monitor for Use of Reputation List on AWS . . . 289
10.3 Server Transaction limits . . . 289
10.3.1 Setting proper limits . . . 291
10.4 Advanced Load Balancing . . . 291
10.5 New restify CDR process . . . 291

11 Reference of Actions 292
11.1 SIP Mediation . . . 292
11.2 SDP Mediation . . . 298
11.3 Monitoring and Logging . . . 302
11.4 Traffic Shaping . . . 303
11.5 Media Processing . . . 305
11.6 SIP Dropping . . . 307
11.7 Scripting . . . 307

11.8 Register Processing . . . 308
11.9 External Interaction . . . 308
11.10 NAT Handling . . . 309
11.11 Other . . . 309
11.12 Default Audio Files . . . 309
11.12.1 Meet-me set PIN audio prompts . . . 309

12 Reference of Global Configuration Parameters **311**
12.1 AWS Parameters . . . 312
12.2 Backup Parameters . . . 312
12.3 CDR Parameters . . . 313
12.4 Event Parameters . . . 313
12.5 Eventbeat Parameters . . . 315
12.6 Firewall Parameters . . . 315
12.7 LDAP Parameters . . . 317
12.8 Lawful Interception Parameters . . . 319
12.9 Login . . . 320
12.10 Low-level Parameters . . . 320
12.11 Miscellaneous Parameters . . . 322
12.12 System Monitoring Parameters . . . 322
12.13 System Monitoring LDAP access Parameters . . . 323
12.14 PCAP Parameters . . . 323
12.15 SEMS Parameters . . . 324
12.16 SIP Parameters . . . 330
12.17 SRTP Parameters . . . 330
12.18 Conference Parameters . . . 331
12.19 Syslog Parameters . . . 331
12.20 Signaling SSL . . . 332

13 Reference of Log Level Parameters **333**
13.1 Debug log level per node or per system . . . 335
13.1.1 Per system . . . 335
13.1.2 Per node . . . 335

14 Reference of Call Agent Configuration Parameters **336**
14.1 Destination Monitor Parameters . . . 336
14.2 Blacklisting Parameters . . . 336
14.3 Registration Agent Parameters . . . 337
14.4 Topology Hiding Parameters . . . 337
14.5 Firewall Blacklisting Parameters . . . 337

15 Reference of Default Port Numbers **338**

16 Reference Interface Parameters **340**

17 Reference Application Interface Options **341**
17.1 SSH . . . 342
17.2 Management GUI . . . 342
17.3 Configuration pull . . . 342
17.4 Media . . . 342
17.5 Signaling . . . 343
17.6 WebSocket signaling . . . 343
17.7 SNMP . . . 343
17.8 XMLRPC . . . 343
17.9 TryIt webrtc client GUI . . . 344
17.10 TURN server for websocket . . . 345
17.11 PCAP query service . . . 345
17.12 Local monitoring query service . . . 346
17.13 Management for host . . . 346

17.14 Local webconf API . . . 346
17.15 HTTP proxy . . . 346
17.16 HTTP redirect . . . 347
17.17 Call state HA replication . . . 347

18 Command Line Reference **348**
18.1 Configuration Management . . . 349
18.2 User Management . . . 349
18.3 Low-Level CLI . . . 350
18.4 HA CLI . . . 350
18.5 ABC Monitor Configuration Management . . . 351

19 Reference of Used Open-Source Software **352**

20 Reference Userdata Parameters for AWS Instances **355**

21 Reference XML-RPC functions **356**
21.1 Provisioned Tables . . . 357
21.2 Call agents . . . 358
21.3 TLS profiles . . . 358
21.4 Nodes . . . 358
21.5 Logical interfaces . . . 358
21.6 System interfaces . . . 358

22 Glossary **359**

Index **361**

Chapter 1

About the ABC Session Border Controller

This manual is a complete handbook for the ABC Session Border Controller (ABC SBC). It documents network planning, SBC installation, policy configuration and the best current practices for operating the SBC.

The ABC Session Border Controller (ABC SBC) is a SIP Back-2-Back User Agent (B2BUA) that provides operators and enterprises with a scalable session border control solution for secure connections with Voice over IP (VoIP) operators and users. With the ABC SBC VoIP service providers and enterprises deploy a session border controller that is designed to run on top of high end hardware as well as appliances and virtual machines. Thereby, the ABC SBC enables VoIP providers to gradually scale up their infrastructure and covers the needs of enterprises of all sizes.

The ABC SBC provides the following features:

- Infrastructure Security: The ABC SBC serves as the first line of defence, fending off attacks coming over the Internet, hiding internal topology, applying rate limits and performing Call Admission Control, limiting number of parallel calls and call length, and off-loading registrar and registration throttling.
- Confidentiality. The ABC SBC implements cryptographic protocols TLS and SRTP that make it incredibly hard for unauthorized third parties to intercept VoIP calls. Secured telephony is possible even without exotic telephones using off-the-shelf webRTC browsers (See the next point). The ABC SBC can also combine cryptographically secured RTC telephony with traditional policy-based IT practices like VPNs, so that confidentiality can be achieved in a practicable end-to-end way between all kinds of equipment.
- Browser Telephony. The ABC SBC includes a built-in SIP/WebRTC gateway. The gateway allows users to interconnect WebRTC browser telephony with SIP telephony and even PSTN telephony users behind SIP/telephony gateways. The browser telephony allows for easy integration with web applications and provides a level of privacy previously unprecedented before in fixed and mobile networks.
- Network Functions Virtualization (NFV). The ABC SBC also comes in a virtualized form that allows administrators to run the SBC without managing the physical infrastructure. More than that, a whole auto-scaling load-balanced RTC gateway cluster can be started in Amazon Elastic Cloud by a single button using the Cloud Formation launching facility. Such a cluster adapts to network conditions, growing and shrinking with network traffic. It can be geographically dispersed for best QoS worldwide and it can be launched in less than five minutes – compare that to the effort of placing your own equipment in multiple geographically distributed air-conditioned data-centers!
- Mediation: The ABC SBC connects disconnected unroutable networks and VLANs, different transport protocols, secured and plain RTP, facilitates NAT traversal, steers codec negotiation, translates identities and adapts SIP headers and bodies for best interoperability between incompatible devices and networks policies.
- Rapid IT integration. The ABC SBC dramatically reduces the time-to-deploy. Studies show that in the vast majority of new network deployments inadequate time and cost is spent in designing data integration concepts. ABC SBC reduces the time-to-deploy with its built-in integration capabilities. Administrators can place external logic to web-servers and govern how the SBC behaves through a RESTful interface. Large amounts of pre-provisioned data can be used to govern the SBC logic, such as routing tables, peering characteristics, SIP bulk registration, blacklists or whitelists, or subscriber information.

- SIP Routing: The ABC SBC's competitive design allows administrators to route SIP traffic based on any message element. Routing methods like source-based, destination-based, least-cost-route based, proprietary-header-field-based and others can be easily configured and cascaded behing each other to find the most-proper destination for SIP traffic.
- Real-time monitoring. The ABC SBC allows its administrator to permanently know what is going on in their SIP networks. Due to the centralized nature of SBCs, the ABC SBC enables you to gain deep insight into the traffic it steers and constantly reports on it using "events" and "Call Detail Records" (CDRs). This data can be further used to perform troubleshooting, backwards analysis and future predictions of the system as whole as well as that of its individual users. The ABC SBC also reports on its status using SNMP.
- Media processing: The ABC SBC includes built-in audio recording, transcoding, announcements and conferencing.
- Web management. Remote management allows rapid and convenient adaptation to ever changing network conditions. ABC SBC's policies can be easily changed through the web interface.
- Non-stop service. The ABC SBC is designed to provide high-availablity by running in redundant hot-standby pairs. Alternate route definitions and built-in monitoring conceal scheduled and unplanned outages of network elements behind the ABC SBC.

1.1 How to Start?

This book is intended for everyone interested in installing and using the ABC SBC. Knowledge of SIP, RTP and IP networking is of an advantage and would ease the reading and use of the book. Of essential value is, however, a good understanding of the VoIP environment in which the ABC SBC is to be deployed. Depending on your goal, there are these options for how to get the most out of this book in the shortest time:

- Cloud RTC Trial: Trialing the RTC gateway using amazon Elastic Cloud allows you to start the WebRTC/SIP gateway service within minutes and establish connectivity between web browsers and an existing SIP service. See the Section *Amazon Elastic Cloud Configuration Cookbook* and visit our trial site at https://go.frafos.com/.
- Installing the ABC SBC: Before installing the ABC SBC it is advisable to go through the practical guide to have a better understanding of the needed infrastructure. After installing the ABC SBC, the practical guide can be used for a quick configuration of the solution. In case certain issues need to be solved that are not covered in the guide, then a look into the reference chapter (Section *Reference of Actions*) will be helpful. The administrator should also go through the administration, monitoring and security chapters to develop a better understanding and control of the installed system.

The book is structured in the following parts:

- *Introduction*: This section provides an overview of the basic technologies addressed here, namely SIP and SBC. Furthermore, the basic concepts and terminology of the ABC SBC are described. If you are knowledgeable with SIP and VoIP deployments you can skip the introductions to SIP and SBCs.
- *Practical Guide to the ABC SBC*: This section provides first an overview of what a future user of the ABC SBC - or actually any other SBC as well - must consider before purchasing and installing an SBC. Moreover, this guide can be seen as a short cut for configuring and using the main features of the ABC SBC without having to go through the entire manual.
- *Installing the ABC SBC*: This section covers the steps needed to deploy the ABC SBC. Firstly, one needs to determine whether to do a complete installation from the FRAFOS repository or to use ABC SBC container version.
- *General ABC Configuration Guide*: This section provides the details about the different features of the ABC SBC, what they are good for and how they can be used. For the more complex parts, additional sections with examples are provided. These example sections are intended as short cuts for solving common issues.
- *ABC SBC System administration*: This chapter explains a set of features available for the administrator of the ABC SBC. These features include the capability to create and manage users of the ABC SBC and define

their rights, the list of commands that can be used to run certain tasks that might not be available through the GUI as well as conduct upgrades and updates of the ABC SBC software.

- *Monitoring and Troubleshooting*: The ABC SBC collects various measurement values and call traces and generates alarms and SNMP traps. These features provide the administrator of the ABC SBC with the information needed to detect errors and problems in the processed VoIP traffic as well as in the operation of the ABC SBC.
- *Securing SIP Networks using ABC SBC and ABC Monitor (optional)*: This chapter provides an overview of the security capabilities of the ABC SBC as well as a guide for configuring blacklists, traffic shaping and limiting the call duration.

1.2 Credits

The initial version of this book was written by the FRAFOS team with support from Sipwise in a three day Book Sprint facilitated by Barbara Rühling. The illustrations were provided by Juan Camilo Cruz. This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License .

Chapter 2

Release Notes

2.1 Release Notes for ABC SBC version 4.6

Note that on major ABC SBC upgrade, the internal db holding call state can be restarted and calls possibly lost, so it is recommended to either perform the upgrade during maintenance window, or if running in HA perform it on non-active node.

Important new features and changes:

- Starting with 4.6 release, there are no default usernames for GUI or config pull access to configuration master node created during ABC SBC installation. They have to be chosen and entered by admin when performing initial configuration.
- On "CCM" container of configuration master it is now mandatory to perform the initial configuration the same way as on standard SBC, as it now requires entering usernames and passwords for GUI and config pull access to CCM.
- On container ABC SBC version, the root user account is now forced to change password on first login.
- On container ABC SBC version under Secuwall, the restore operation does not restore system files.
- It is now possible to configure TURN server using "App" on interface in SBC GUI. New coturn version used, which includes security fixes.
- Newer version of net-snmp 5.8 used.
- The SBC CCM GUI now offers possibility to check configuration before its activation, which also allows to see a list of SBC nodes that will be affected by the new configuration.
- If a bundled certificate is uploaded in CCM GUI then all bundled certificates are displayed for a given TLS profile.
- A support for TLS profile related template parameter was added. It can be used while configuring interfaces to specify a TLS profile.
- A support for direct assignment of Call Agents to one or more nodes was added.
- If a template parameter for a Call Agent is declared, it is possible to use Call Agents picker either during specifying a new variable or from config group/node directly.
- If a template parameter for a TLS profile is defined, it is possible to use TLS profile picker either during specifying a new variable or from config group/node directly.
- A new configuration was added into CCM GUI for an eventbeat service
- Fixed UTF support in CDR logs
- A webconference application supports a mode where room and pin are entered separately
- The GUI offer 2 new interfaces applications: *redirect* and *proxy*

- API now host swagger redoc documentation on */api(/v{1,2,3}/docs/redoc*
- SEMS offer the possibility to blacklist some IP without time limit
- Better support for diacritics characters
- Proper SIP mapping to Q850
- Introduction of a global "debug" mode, toggling for all ABC SBC services more verbose logging
- TLS profiles, nodes and interfaces could be now managed via XML-RPC
- Logical interfaces now can be renamed via CCM GUI
- SNMP configuration has been moved under Global config in CCM GUI
- The vmware virtual image of SBC is no more provided. For trial purposes, it is recommended to use container version.
- The SBC selinux policy, which was in experimental state and didn't work, was removed. The selinux is required to be disabled and gets disabled on ABC SBC installation.
- On each ABC SBC upgrade, a full sync of provisioned tables data from CCM to SBC nodes is forced.
- It is possible to use routing rules per node, not just a config group.
- It is possible to edit a node hostname.
- It is possible to clone a call agent.
- GUI shows commands for initial configuration of nodes if this is needed.
- "Replace header value" action now also acts on To & From headers.
- SDP manipulation was introduced.
- The system accounts password files were removed from sbc-backup and sbc-restore, as too dangerous.

For a detailed list of all changes and fixes, please check the SBC changelog, which is shown at the beginning of the upgrade process or can be found in /usr/share/doc/frafos-sbc/CHANGELOG.md file on the node after the upgrade.

2.2 Release Notes for ABC Monitor version 4.6

Important new features and changes:

- There is no more default password for the administrative user "sbcadmin" used for ABC Monitor GUI access, and it has to be set using "abc-monitor-password" from command line before first GUI login.

IMPORTANT: It has to be set also in case of upgrade to 4.6, in case the default password was used unchanged before. The upgrade will ask for setting the password, if running interactively.

- On container ABC Monitor version, the root user account is now forced to change password on first login.
- Added support for LDAP authentication to Monitor GUI. It can be configured on Monitor GUI Settings page, or alternatively (if GUI is not accessible yet) using "abc-monitor-ldapconfig" command.
- On container ABC Monitor version, the default GUI https port is 445.
- Number of shards is now calculated based on configured number of days to keep old data. This is configurable from GUI
- Call variables are now stored in "var" (not in attrs) and they are searchable automatically.
- Added a new graph "AVG MoS" which presents MoS average values.
- Added charts animations
- Added a new alarm for dead CA
- Added a possibility to download a pcap sequence diagram from call events

- Alarms are disabled by default

For a detailed list of all changes, please check ABC Monitor changelog, which is shown at the beginning of the upgrade process or can be found in /usr/share/doc/frafos-monitor/CHANGELOG-Monitor.md file on the Monitor after the upgrade.

Chapter 3

Introduction

3.1 A Brief Introduction to History and Architecture of SIP

The Session Initiation Protocol (SIP, **RFC 3261**) is a backbone of every VoIP network nowadays. Its "language" is used by telephony devices to find each other, signal who is calling whom, negotiate which audio/video codecs to use and even more. The telephony devices are typically SIP desktop phones, but it may be also smartphones, softphones, or massive PSTN gateways with PSTN infrastructure and users behind them. In between, there are intermediary SIP network devices that help to locate the end-user devices, perform Call Admission Control, help often with various imperfections of the end-devices and perform other useful functions. The ABC Session Border Controller is one of such, however other kinds of SIP proxy servers, Back-to-Back User Agents, specialized application servers, and more are common.

VoIP began to reach market back in mid nineties. By then Internet had established itself as a consumer product. The number of users buying PCs and subscribing to an Internet Service Provider (ISP) for a dial-up access was increasing exponentially. While mostly used for the exchange of Email, text chatting and distribution of information VoIP services based on proprietary solutions as well as H.323 started to gain some popularity. The standardization organization Internet Engineering Task Force, IETF, began to devise its own protocol suite. Some protocols existed already by then. The Real-Time Transport Protocol (RTP) **RFC 1889** enabled the exchange of audio and video data. The Session Description Protocol (SDP) **RFC 2327** enabled the negotiation and description of multimedia data to be used in a communication session. The first applications, often open-source, for sending and receiving real-time audio and video data emerged. A signaling protocol was missing, however.

In those days, the procedure for establishing a VoIP call between two users based on the IETF standards would look as follows: The caller starts his audio and video applications at a certain IP address and port number. The caller then either calls the callee over the phone or sends him an Email to inform him about the IP and port address as well as the audio and video compression types. The callee then starts his own audio and video applications and informs the caller about his IP and port number. While this approach was acceptable for a couple of researches wanting to talk over a long distance or for demonstrating some research on QoS this was clearly not acceptable for the average Internet user.

The Session Initiation Protocol (SIP) **RFC 3261** was the attempt of the IETF community to provide a signalling protocol that will not only enable phone calls but can also be used for initiating any kind of communication sessions. SIP has been contemplated for use by audio and video calls, as well as for setting up a gaming session or controlling a coffee machine.

The SIP specifications describe three types of components: user agents (UA), proxies and registrar servers. The UA can be the VoIP application used by the user, e.g., the VoIP phone or software application. A VoIP gateway, which enables VoIP users to communicate with users in the public switched network (PSTN) or an application server, e.g., multi-party conferencing server or a voicemail server are also implemented as user agents. The registrar server maintains a location database that binds the users' VoIP addresses to their current IP addresses. The proxy provides the routing logic of the VoIP service. When a proxy receives a SIP request from a user agent or another proxy it also conducts service specific logic, such as checking the user's profile and whether the user is allowed to use the requested services. The proxy then either forwards the request to another proxy or to another user agent or rejects the request by sending a negative response.

While the server roles prescribed by the SIP specification are functional, actual implementations found in networks tend to integrate multiple roles in a server product. A registrar is often co-located with a proxy server so that they can share user-location databases. A server can also present itself as User-Agent to both sides of a signaling session to be able to manipulate SIP messages more extensively than the proxy specification would permit.

Every signaling SIP transaction consists of a request and one or more replies. The three most commonly used request types are REGISTER, INVITE and BYE. The REGISTER request makes a SIP phone's address known to a SIP server so that it knows where to forward incoming SIP requests. The INVITE request initiates a dialog between two users. A BYE request terminates this dialog. Responses can either be final or provisional. Final responses can indicate that a request was successfully received and processed by the destination. Alternatively, a final response can indicate that the request could not be processed by the destination or by some proxy in between or that the session could not be established for some reason. Provisional responses indicate that the session establishment is in progress, e.g. the destination phone is ringing.

In general one can distinguish between three types of SIP message exchanges, namely registrations, dialogs and out of dialog transactions.

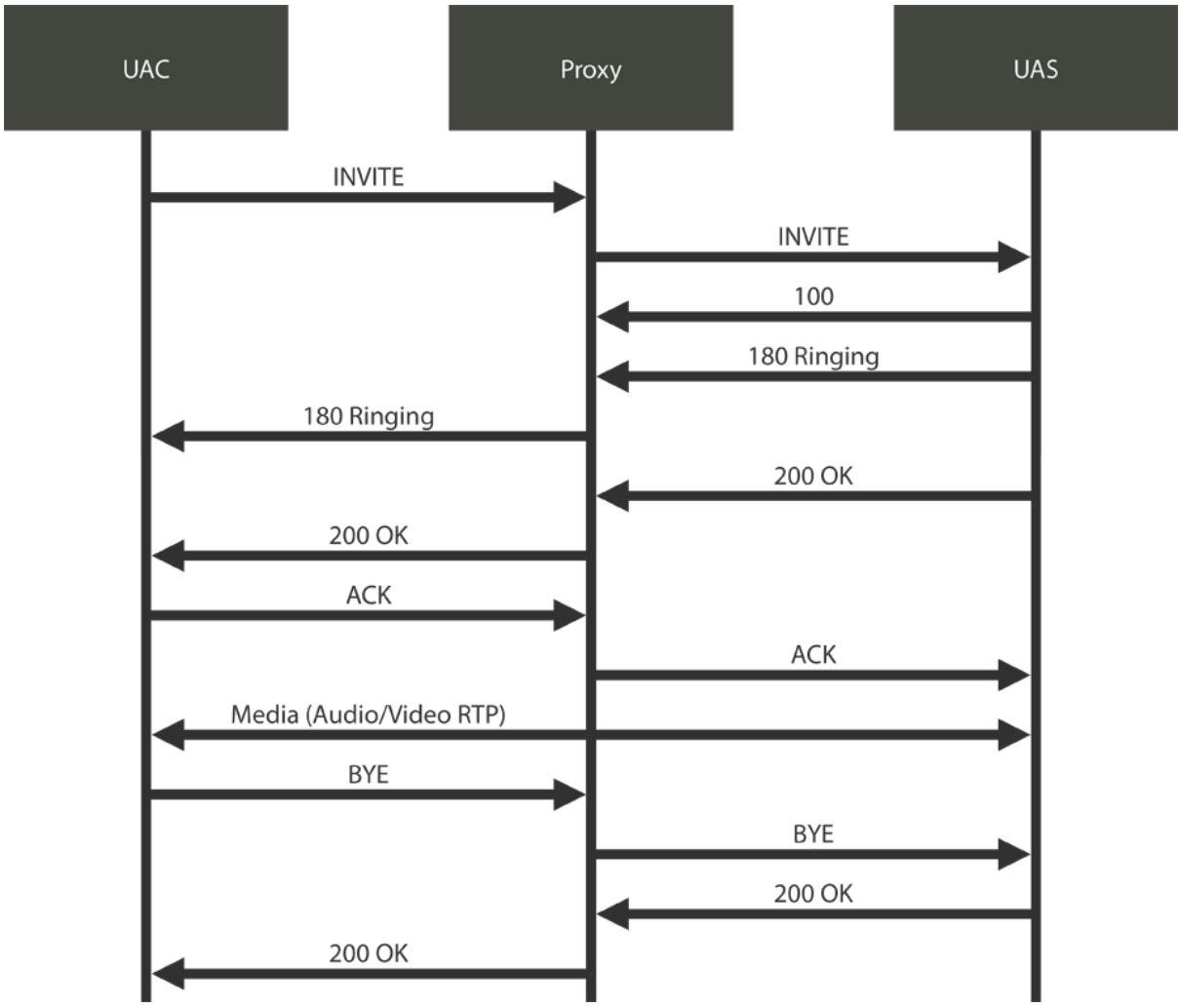


Fig. 1: SIP Call flow

A SIP registration enables a user agent to register its current address, IP address for example, at the registrar. This enables the registrar to establish a correlation between the user agent's permanent address, e.g. sip:user@frafos.com, and the user agent's current address, e.g., the IP address used by the user's user agent. In order to keep this correlation up to date the user agent will have to repeatedly refresh the registration. The registrar will delete a registration that is not refreshed for a while.

A SIP dialog, a call for example, usually consists of a session initiation phase in which the caller generates an

INVITE that is responded to with provisional and final responses. The session initiation phase is terminated with an ACK, see *SIP Call flow*. A dialog is terminated with a BYE transaction. Depending on the call scenario the caller and callee might exchange a number of in-dialog requests such as reINVITEs or REFER.

The last type of SIP interactions is SIP transactions that are not generated as part of a dialog. Examples of out of dialog SIP requests include OPTIONS and INFO that are often used for exchanging information between SIP nodes or as an application level heartbeat.

Every SIP message consists of three parts: First line, message header and message body, see *Content of SIP messages*. The first line states the purpose of the message. For requests it identifies its type and the destination address. For replies the first line states the result as a numerical 3-digit status code together with a textual human-readable form. The second part of the message, the header part, includes a variety of useful information such as identification of the User Agent Client and the SIP path taken by the request. The third part includes a message body that contains application specific information. This can be for example session description information (SDP) indicating the supported codecs.

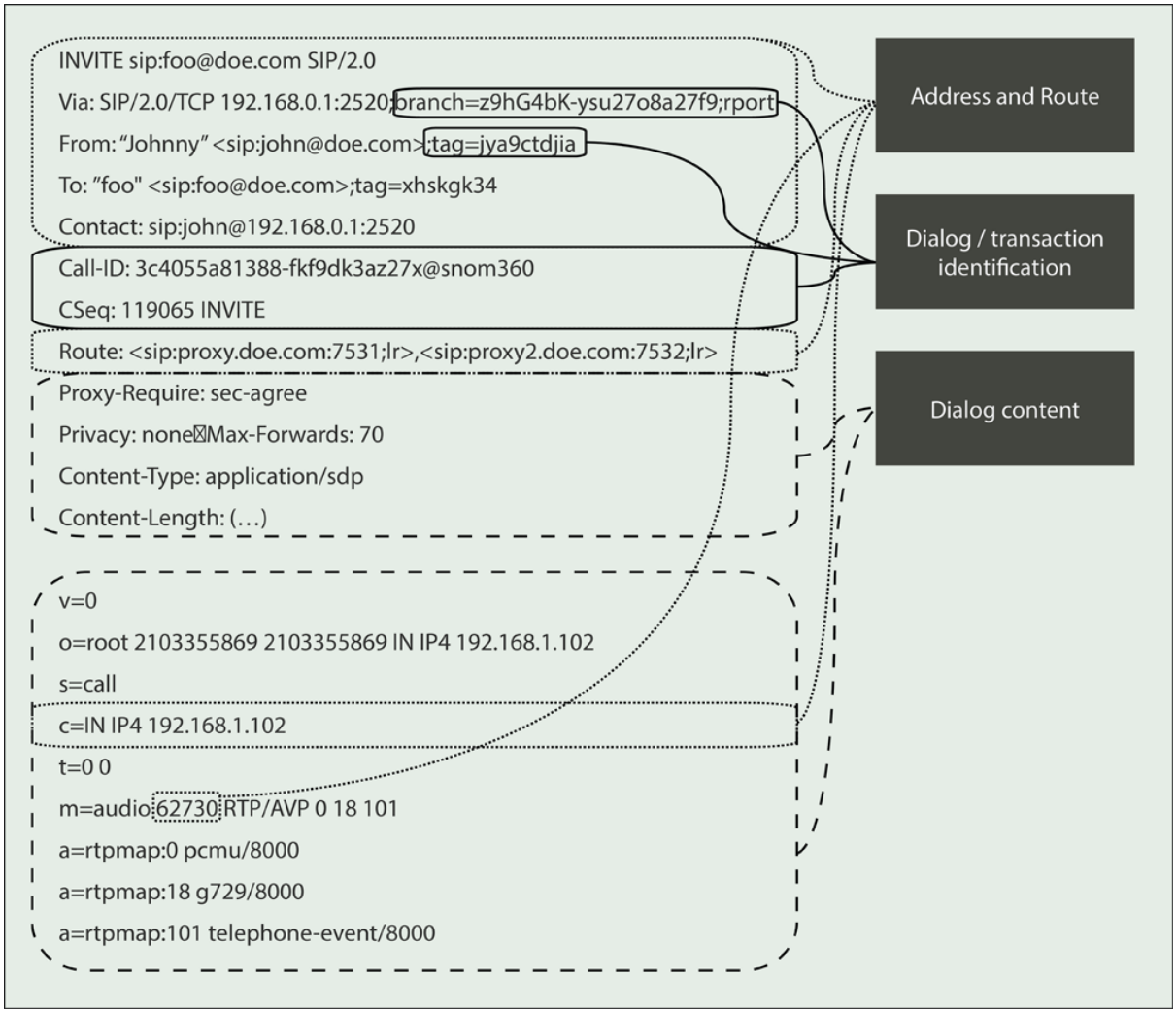


Fig. 2: Content of SIP messages

The information contained in these three parts can be roughly divided into three categories, see *Content of SIP messages*:

- Addressing and routing information: This includes information about who has sent the message and where it is destined to, the next hop to be sent to as well as the hops it has traversed. This information is included in the first line as well as in different headers such as From, To, Contact, P-Asserted-Identity, Via, Route, Path and other headers. The message body can contain information about where the media traffic should be sent to or is expected to come from.

- Dialog and transaction identification: This part of a SIP message is used to uniquely identify a SIP dialog or transaction. This information is included in SIP headers such as Cseq, Call-Id and tags included in From, To and Via headers.
- Dialog content: With dialog content we categorize data that is included in a SIP message that is either used to describe certain features of a dialog or indicates how a node receiving the message should process the message. This can include parts of the SIP message body carrying SDP, which includes description about which audio or video codes to use. Certain headers such as Privacy for example indicate the user's wishes with regard to the way private information such as user address should be dealt with.

3.2 What is a Session Border Controller (SBC)?

Historically Session Border Controllers emerged after publication of the SIP standard as a panacea to early protocol design mistakes: ignorance of Network Address Translators (NATs), unclear data model, liberal syntax, reluctance to standardize legal interception and more.

Probably the single biggest mistake in the design of SIP was ignoring the existence of network address translators (NAT). This error came from a belief in the IETF leadership that IP address space would be exhausted more rapidly and would necessitate global upgrade to IPv6 which would eliminate the need for NATs. The SIP standard has assumed that NATs do not exist, an assumption, which turned out to be a failure. SIP simply didn't work for the majority of Internet users who are behind NATs. At the same time it became apparent that the standardization life-cycle is slower than how the market ticks: SBCs were born, and began to fix what the standards failed to do: NAT traversal.

Yet another source of mistakes has been the lack of a clear data model behind the protocol design. Numerous abstract notions, such as dialog or session, transaction or contact simply didn't have unique unambiguous identifiers associated with them. They were calculated or almost guessed out of various combinations of header-fields, decreasing the interoperability. Some message elements, such as *Call-ID*, have been overloaded with multiple meanings. While some of these were fixed in the later SIP revision and its extensions (*rport* **RFC 3581**, *branch*, gruu **RFC 5628**, session-id) the market forces jumped in quickly. SBCs began to implement "protocol repair".

The other class of mistakes emerged from implementations. Many SIP components were built under a simplifying assumption that security comes for free. Numerous implementations were found to be vulnerable to malformed SIP messages or excessive load. The SBCs began to play a security role.

The reality in today's real time communication networks is that, contrary to the end-to-end design of the Internet and its protocols, service operators can achieve the best user experience by exerting tight control - over the endpoints and over the interface to peering networks.

Over several years, Session Border Controllers became a de facto standard for which ironically no normative reference existed. A non-normative information reference on the subject, **RFC 5853** was published as late as in 2013. Session Border Controllers nowadays handle NATs, fix oddities in SIP interoperability and filter out illegitimate traffic. They began to incorporate elements of the standardized SIP components. For example, routing functionality contemplated by the standards for proxy servers is nowadays part of SBC products. Similarly the SBCs often incorporate media recording and processing functions, whether that's for quality assurance, archiving or legal-compliance purposes.

3.2.1 General Behavior of SBCs

Purist SIP call flow depicts the message flow of an INVITE request between a caller and a callee. This is the simplest message sequence that one would encounter with only one proxy between the user agents. The proxy's task is to identify the callee's location and forward the request to it. It also adds a *Via* header with its own address to indicate the path that the response should traverse. The proxy does not change any dialog identification information present in the message such as the tag in the *From* header, the *Call-Id* or the *Cseq*. Proxies also do not alter any information in the SIP message bodies. Note that during the session initiation phase the user agents exchange SIP messages with the SDP bodies that include addresses at which the agents expect the media traffic. After successfully finishing the session initiation phase the user agents can exchange the media traffic directly between each other without the involvement of the proxy.

SBCs come in all kinds of shapes and forms and are used by operators and enterprises to achieve different goals. Actually even the same SBC implementation might act differently depending on its configuration and the use case. Hence, it is not easily possible to describe an exact SBC behavior that would apply to all SBC implementations. However, in general one we can still identify certain features that are common for most of SBCs. For example, most SBCs are implemented as "Back-to-Back User Agent" (B2BUA).

A B2BUA is a proxy-like server that splits a SIP transaction in two pieces: on the side facing the User Agent Client, it acts as server; on the side facing the User Agent Server it acts as a client. While a proxy usually keeps only state information related to active transactions, B2BUAs keep state information about active dialogs, e.g., calls. That is, once a proxy receives a SIP request it will save some state information. Once the transaction is over, e.g., after receiving a response, the state information will soon after be deleted. A B2BUA will maintain state information for active calls and only delete this information once the call is terminated.

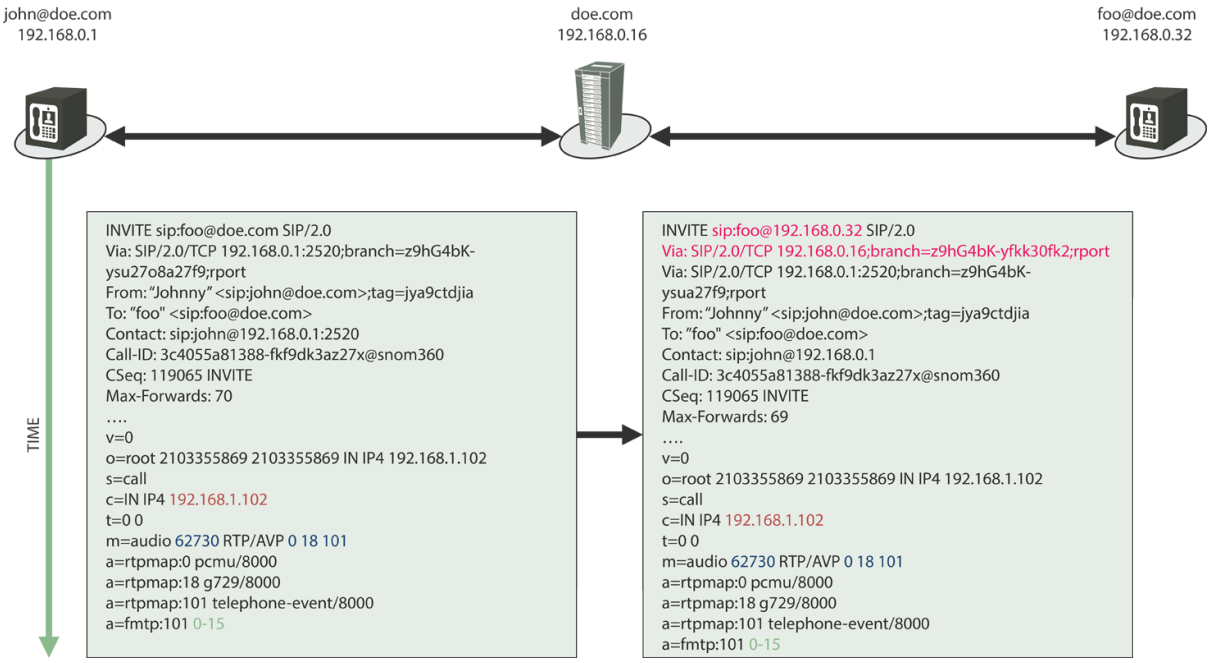


Fig. 3: Purist SIP call flow

SIP call flow with SBC depicts the same call flow as in *Purist SIP call flow* but with an SBC in between the caller and the proxy. The SBC acts as a B2BUA that behaves as a user agent server towards the caller and as user agent client towards the callee. In this sense, the SBC actually terminates that call that was generated by the caller and starts a new call towards the callee. The INVITE message sent by the SBC contains no longer a clear reference to the caller. The INVITE sent by the SBC to the proxy includes *Via* and *Contact* headers that point to the SBC itself and not the caller. SBCs often also manipulate the dialog identification information listed in the *Call-Id* and *From* tag. Further, in case the SBC is configured to also control the media traffic then the SBC also changes the media addressing information included in the *c* and *m* lines of the SDP body. Thereby, not only will all SIP messages traverse the SBC but also all audio and video packets. As the INVITE sent by the SBC establishes a new dialog, the SBC also manipulates the message sequence number (*CSeq*) as well the *Max-Forwards* value.

Note that the list of header manipulations listed in *SIP call flow with SBC* is only a subset of the possible changes that an SBC might introduce to a SIP message. Furthermore, some SBCs might not do all of the listed manipulations. If the SBC is not expected to control the media traffic then there might be no need to change anything in the SDP lines. Some SBCs do not change the dialog identification information and others might even not change the addressing information.

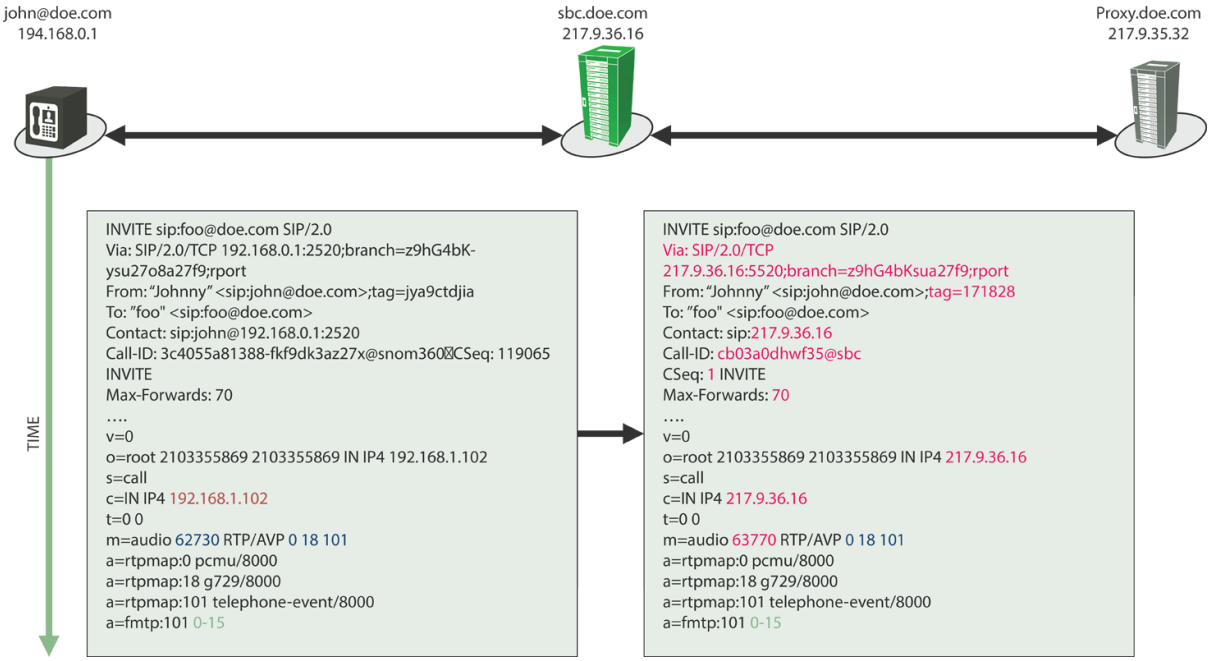


Fig. 4: SIP call flow with SBC

3.2.2 General Deployment Scenarios of SBCs

Session border controllers are usually deployed in a similar manner to firewalls, namely with the goal of establishing a clear separation between two VoIP networks.

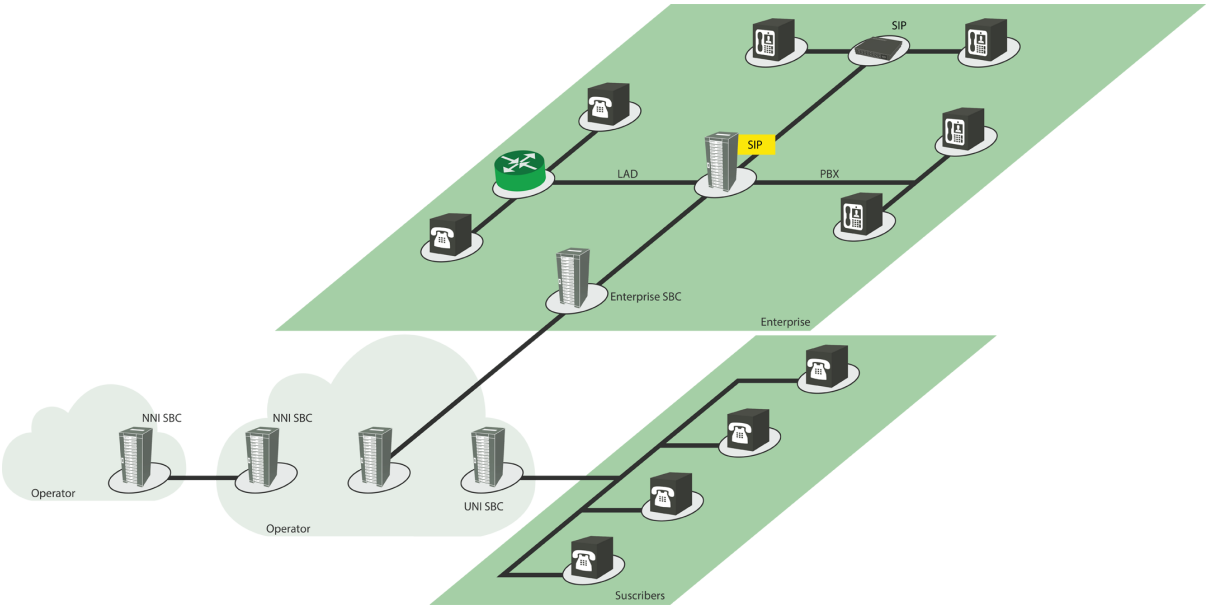


Fig. 5: SBC deployment scenarios

In general one can distinguish three deployment scenarios, see *SBC deployment scenarios*:

- User-Network-Interface (UNI): Operators use SBCs to establish a secure border between their core VoIP components and subscribers. The core components consists of PSTN gateways, media servers, SIP proxy and application servers. Subscribers use SIP hardphones and softphones, Internet Access Devices that connect analog and digital phones to the IP network, and newly web browsers deploying the WebRTC standard. Most important administrative tasks in this scenario include facilitation of NAT traversal (see *NAT*

Traversal), achieving interoperability among multiple types of clients (see *SIP Mediation*), security against attacks coming from the public Internet (see *Securing SIP Networks using ABC SBC and ABC Monitor (optional)*) and off-loading registrar (see Section *Registration Caching and Handling*).

- Network-Network-Interface (NNI): In the NNI interface, two operators connect to each other directly over SIP. Most important administrative concerns in this scenario include mediation of different network policies (see *SIP Mediation*), enforcement of service-level-agreements between providers by traffic shaping (see *Traffic Limiting and Shaping*) and multi-provider SIP routing (see *SIP Routing*).
- Enterprise SBC (E-SBC): Enterprises are increasingly replacing their PBXs with VoIP PBX or are extending their PBX with a VoIP module to benefit from attractive VoIP minute prices. Enterprise SBCs are used to secure the access to the PBX. The enterprise SBC is also expected to secure the communication to the VoIP operator, which is offering the VoIP service to the enterprise. Typical administrative concerns include harmonization of dialing plans between an enterprise and its trunking partners using the mediation feature see *SIP Mediation*), and setting up secured VoIP connectivity for web-users (see *Securing SIP Networks using ABC SBC and ABC Monitor (optional)*).

3.3 Do You Need an SBC?

Before installing an SBC it might be worth thinking whether an SBC is needed in the first place. To answer this, here are a couple of questions:

- will you deal with SIP devices behind a NAT? If the answer is yes then deploying an SBC is most likely the right choice. While there are already a number of NAT traversal solutions such as STUN **RFC 5389**, TURN **RFC 5766** or ICE **RFC 5245** these solutions either do not solve all issues or require certain extensions at the end devices which are not always available.
- do you deploy SIP devices that you would not want other users or operators to be able to send SIP or media traffic directly to? This is usually the case when a PBX or a PSTN gateway is deployed. If the answer is yes then an SBC would be the right choice as it would hide the IP addresses of these devices and prevent direct communication to them.
- do you deploy a heterogenous set of VoIP devices? If yes then an SBC can be the proper point in the network to fix interoperability issues by normalising the traffic and solving issues created by protocol implementation peculiarities.
- do you want to protect your VoIP devices from Denial of Service attacks? If there is the danger that an attacker might overload your network and VoIP devices by generating a large amount of SIP requests and RTP packets then an SBC would act as a first line of defence and filter the malicious traffic before it reaches the core VoIP components.
- do you want to reduce the possibilities of fraud? If there is a danger that a fraudulent user might try to make more calls than allowed then an SBC would be the best approach. With an SBC it is possible to reliably limit the number of calls made by a user.
- do you want to protect your users from a bill shock? When a user calls an expensive number and fails to terminate the call in a proper manner then he will most likely get a shock when receiving the bill for a call lasting for hours. An SBC on the border of the network can be configured so as to cut calls after a certain period of time and hence limit the damage.
- will you need to transcode the media? If different users are using different codecs - which is especially the case when connecting mobile to fixed networks - then media transcoding will be needed. Media transcoding is often an integral part of SBCs.
- will your users be using browser telephony using WebRTC? Then you need to connect them to the rest of SIP world and SIP-PSTN gateways using the built-in WebRTC gateway.

3.4 ABC SBC Networking Concepts

This section provides an overview of the main concepts and terms of the ABC SBC. It shows the overall model of SBC-managed networks, how the SBC connects to the individual networks using "Interfaces", models SIP devices as "Call Agents", and groups these in "Realms". Eventually A-B-C rules are described that define how the ABC SBC manages SIP traffic as it passes through it.

3.4.1 Network Topology

As depicted in the Figure *ABC SBC Concepts Overview*, the ABC SBC communicates with VoIP phones, media servers and other entities that act as SIP user agent. We call these entities Call Agents (CA) and group them into so called Realms. The ABC SBC associates rules with Call Agents and Realms. These rules fully describe how every single session traversing the ABC SBC from one CA/Realm to another is processed.

The rule processing occurs in three steps. When receiving a SIP message from a Call Agent, the ABC SBC will first execute inbound rules ("A-Rules") associated with the Call Agent and the Realm it belongs to. These rules typically implement all kinds of admission control. Once the message is accepted the ABC SBC applies routing rules ("B-Rules") to determine the Call Agent where to send the message to. Before actually forwarding the message to the destination, the ABC SBC executes its outbound "C-rules". The C-Rules are typically used to transform the SIP messages to conform to practices used by the destination, such as local specific dialing conventions.

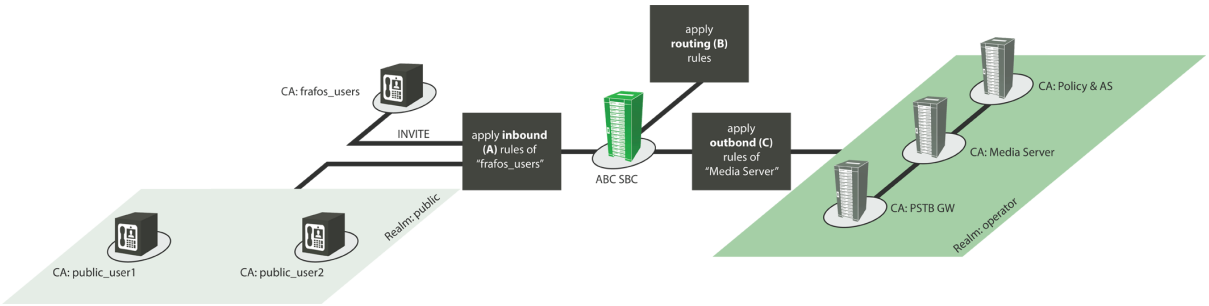


Fig. 6: ABC SBC Concepts Overview

3.4.2 SBC Interfaces

SBC Interfaces define how the ABC SBC connects to the adjacent IP networks. They are are an abstraction layer on top of the network interfaces. Specifically, the SBC Interfaces define through which IP addresses, port numbers and network interfaces the ABC SBC offers its services.

There are the following types of SBC interfaces:

- Internal management (IMI): used in high availability (HA) and cluster setups as the communication channel between the SBC node servers.
- External management (XMI): used for maintenance access (graphical user interface, ABC Monitor access, etc. . .)
- Media (MI): used for receiving and sending media payload.
- Signalling (SI): used for receiving and sending SIP signalling messages.
- Websocket Signaling (WS): used for receiving and sending SIP over websocket from and to WebRTC clients.

Each of the SBC interfaces is mapped to a physical or system network interface that is used for the actual sending and receiving of the data. Multiple SBC interfaces can be mapped to the same network interface. If VLANs are used, they are administered under management of physical interfaces and remain otherwise transparent to the rest of the system.

Administration of the SBC interfaces is described in the Section *Interface Configuration*.

3.4.3 Call Agents

Call Agents (CA) are the smallest type of peering entities the ABC SBC can differentiate. They represent logical end-points. They can be defined based on several addressing mechanisms:

- IP address and port
- Domain or host name and port
- IP network and mask

Additionally, a Call Agent is assigned to a signalling and a media interface. These interfaces are used whenever SIP signalling or media packets are sent to or received from a Call Agent.

For security reasons, the SBC communicates by default only with well-known and defined Call Agents. When an incoming SIP request cannot be attributed to a Call Agent, it is rejected.

To determine the source Call Agent, the SBC uses the source IP address and port of the request to search among the configured Call Agents. If the definitions of Call Agents are overlapping (for example when some Call Agents are defined with an IP address which belongs to a subnet used to define another Call Agent), the following descending order is used to determine the Call Agent:

- Call Agents with matching IP address and port.
- Call Agents with matching IP address but a port equal to 0.
- Call Agents with matching IP network (including mask) in descending order of mask length

Routing rules determine the target Call Agent. In this case, the interface used to send the SIP signalling is the one assigned to the target Call Agent. In case media relay is used, the media interface assigned to this target Call Agent is also used accordingly. The target Call Agent is used to determine the set of applicable rules on the outbound side as well. Note that Call Agents specified by subnet address cannot be used for routing.

3.4.4 Realms

Every Call Agent belongs to one Realm. Realms are the logical groupings of one or more Call Agents. They allow multiple Call Agents to share the same SIP processing logic without defining it individually multiple times.

In a classical context where the SBC is placed on the border of an internal network, it is common to define one Realm for the outside world, and one for the internal network. This way, all the restrictive rules to protect the internal network are defined for the outside Realm, while the internal one can be safely trusted.

In a peering use case, usually one Realm per peering partner is defined.

3.4.5 A-B-C rules

The ABC SBC is fundamentally rules driven. This means that almost all features can be activated based on certain conditions evaluated at run-time, based on parts of the signalling messages or media payload.

All rules are constructed using the same pattern. They consist of a set of one or more conditions. If all conditions apply (logical conjunction), a set of one or more actions is executed.

It is important to understand that rules are generally applied only on dialog-initiating requests or out-of-dialog requests. However, some actions have a scope that goes beyond these dialog-initiating requests or out-of-dialog requests. For example, header filters apply to all requests exchanged, including in-dialog requests. Action descriptions include their scopes where applicable.

There are three types of rules that are always executed in the same order: A, B, and C. A-rules describe how incoming traffic for a Call Agent/Realm is handled, B-rules determine destination for the SIP request, and C-rules describe SIP processing behaviour specific to that chosen destination.

A and C rules are associated with Call Agents and/or Realms. The realm rules allow to have shared logic for all Call Agents that are to be handled the same way, while CA rules are suitable for individual logic. Often, rules are associated with both Realms and Call Agents. The Realm rules are executed first, and their results can be overridden by more specific Call Agent Rules.

B rules are different in that they are global. They are not associated with a specific realm or call agent. When processing of A-rules completes, the B-rules determine the next hop. That's is the only action the B-rules can perform. Then Realm and Call Agent specific C-rules are processed.

The FRAFOS ABC SBC handles calls according to the schema shown in the figure *Call handling algorithm*.

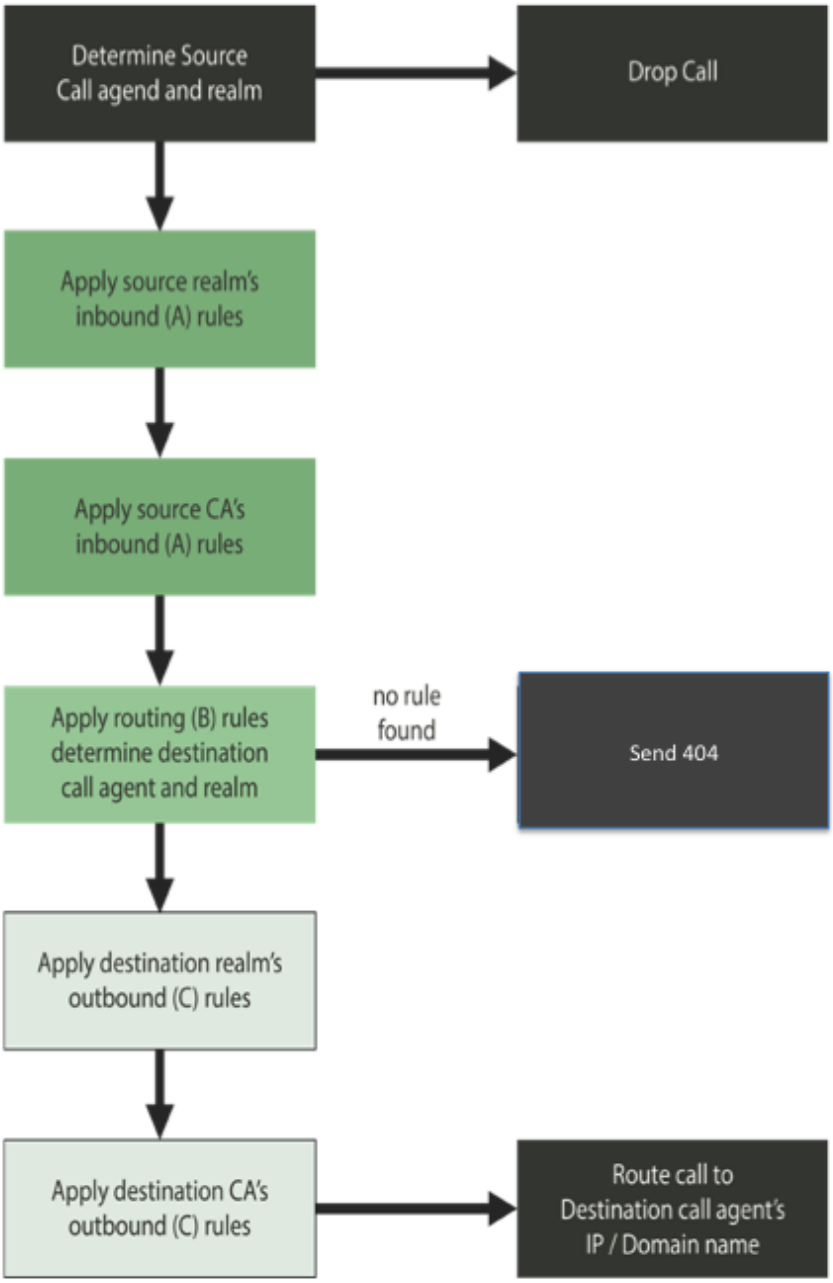


Fig. 7: Call handling algorithm

It is a good practice to place rules in realms rather than in Call Agents, unless they are clearly specific to Call Agents. Rules in realms don't have to be repeated Call Agent by Call Agent and don't expose administrator to Copy-and-Paste administrative errors.

Conditions and Actions

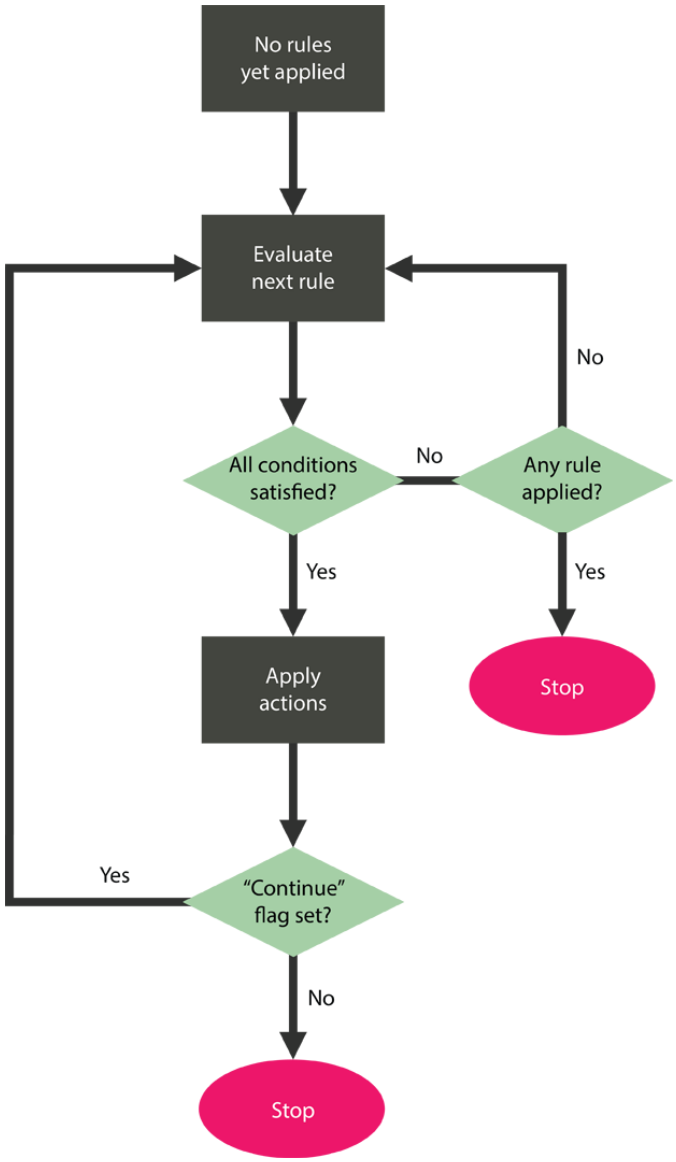


Fig. 8: Rule evaluation sequence

Every A/B/C rule may have one or multiple conditions. The conditions can check incoming message content (method, R-URI, headers, codecs), source (IP, port, realm), values stored in call variables etc.

Rules have zero or more actions. If all its conditions are satisfied ('AND' combination), the actions of a rule are executed in the order in which they are defined. In case a rule does not contain any conditions, the rule's actions are always applied.

Actions can have parameters depending on the action type. For example, the action "Add Header" that appends a SIP header to an outgoing message takes two parameters - a header name and a header value.

Within a rule set (Realm A or C rules; Call Agent A or C rules), the SBC evaluates each rule by evaluating the condition set first. If all conditions match, the set of actions is executed. As part of the rule definition a "continue flag" is defined. If the "continue flag" is checked, the next rule is evaluated. Otherwise, the rule evaluation within

this block stops. Irrespectively of the state of the "continue flag", the rule evaluation continues with the next block. This means that if the "continue flag" is not checked in a Realm A rule where the conditions match, the Call Agent A rule will still be executed, see *Rule evaluation sequence*.

Routing rules

Routing rules have the same set of conditions as A & C rules, but only one possible action: route the request to a target Call Agent.

Fig. 9: B-Rule evaluation

The essential role of B rules is to determine to which target Call Agent the processed request will be sent to. This also influences the outbound signalling and media interface used to send out the forwarded request.

To determine a target Call Agent, the SBC will evaluate the conditions for each routing (B) rule. Once a match is found, the Call Agent associated with the routing rule is determined as the target Call Agent. Then, the processing continues with the C rules assigned to the target Call Agent.

As depicted in Figure *B-Rule evaluation*, all active B rules are traversed and evaluated sequentially. In case all conditions of a rule are satisfied, the destination call agent and routing method are successfully determined. In case that the conditions of a rule are not satisfied, processing continues with the next rule. If no matching routing rule can be found, then the call is refused with a *404 Not Found* error code and an *error* event type is produced.

Chapter 4

Practical Guide to the ABC SBC

4.1 Network Planning Guidelines

This section provides you with a list of steps every network administrator shall walk through carefully before deploying an SBC-powered network. Early planning helps to create robust network that well serves the needs of its users and make administrator's life free of surprises.

Each planning step starts with a question about a particular network planning aspect. Administrators need to to ask themselves this question to determine their configuration needs. It is then followed by a short debate of the most important configuration options and trade-offs. Not all available options are included, yet those present are of major importance and shall be answered in the early planning phase.

The subsequent section includes a checklist that reiterates question raised in the section. We recommend going through it thoroughly before a deployment is commenced, and also completing the answers in the "Customer Site Survey" document available from our customer care.

The steps in this section are grouped as follows:

- The Section *Topology Model* describes how an ABC SBC connects to the IP networks, how it models SIP devices and groups them administratively.
- The Section *SBC Logic* describes the anticipated behaviour of the ABC SBC and what needs to be considered when configuring it: routing, media processing, NAT handling, and more.
- The Section *Security Policies* summarises configuration steps needed to protect both the connected SIP networks and the SBC itself.
- The Section *Capacity planning* provides guidelines for estimating the SBCs cluster dimensions.
- Eventually the Section *IT Integration* discusses the configuration steps that need to be considered if the SBC connects to other network management elements.

4.1.1 Topology Model

The key function of the SBC is to securely connect various SIP elements and peering networks together. This is not trivial because the networks and devices may use different security policies, SIP protocol extensions, dialling plans, codecs, etc. To deal with this variety the SBC uses a network model in which the SIP devices are represented as abstract "Call Agents" that are grouped in "Realms". With Call Agents (CAs) and Realms there are rules associated that characterise how their traffic from and to them is treated.

The topology planning process includes the following steps:

IP layer topology

To which IP networks does the SBC connect?

The first step in defining your topology is located at the IP layer: you have to specify which IP networks connect with each other and how the SBC connects to them. This is captured in the specification of "interfaces".

Interfaces describe in detail how an SBC connects to IP networks. In the simplest case all traffic can be routed through a single Ethernet card using a single IP address and dedicated UDP port range. Many deployments use multiple Ethernet cards or VLANs to connect two or more physical networks with different levels of security.

A widely used practice we recommend is use of three network cards for three networks: unprotected public, protected private, and highly-protected administrative. Then for example residential SIP phones and peering providers connect to the SBC over the public network, operator's PSTN gateways are located in a private IP subnet, and administrators access the SBC over the administrative networks.

Note that using fewer cards makes a clean separation and security of traffic more difficult and also reduces total throughput.

More detailed discussion of interface configuration is described in Section *Physical, System and SBC Interfaces*.

IP layer security

Which firewall rules shall be used in firewalls and the SBC?

Once the IP connectivity is specified, you need to specify L3/L4 restrictions for your deployment. This consists of two parts: if you deploy additional L3/L4 firewalls in front of the SBCs you must make sure that they do not restrict legitimate traffic. You must allow SIP traffic (typically UDP/TCP port 5060), and if media anchoring is used also unprivileged UDP ports. On the SBC you may take the opposite approach and restrict critical ports, especially if no L3/L4 firewall is used. At least you shall make sure that traffic to privileged ports used for administration is permitted only from trusted IP addresses.

More can be found in the Section *Manual IP-layer Blocking* .

Call Agents (CAs)

What SIP Devices does the SBC talk to?

Identify CAs by IP address, IP address range or DNS name. A CA may be physically a PSTN gateway, a whole "cloud" of identical IP phones located in a subnet, a peering party, just anything with unique identification and characteristics. Also specify if there is some specific treatment a CA shall obtain and that needs to be specified in SBC rules. These frequently include:

- traffic limitations, i.e. will you impose one of these constraints on the traffic: RTP bandwidth, signalling rate, number of parallel calls?
- mediation rules, i.e., do you to reconcile dialling plans, identity (URI) usage, header usage? Note that some of these rules cannot be easily planned for ahead of time as they are used to fix protocol imperfections discovered after fact in operation.
- NAT handling, i.e. does presence of NATs necessitate use of media relays, and shall the SBC handle traffic symmetrically? (normally the answer is yes to both these questions if there are NATs present).

More details about traffic limitations are described in the Section *Traffic Limiting and Shaping*, mediation is described further in Section *SIP Mediation* and media anchoring is described in more detail in the Sections *NAT Traversal* and *Media Anchoring (RTP Relay)*.

Realms

What SIP Networks does the SBC talk to?

Most CAs belong to an administrative zone, whose traffic the SBC handles the same way. It would be impractical to define traffic rules for every single CA in such a zone. Therefore the SBC uses the concept of Realms that group all CAs sharing the same characteristics. For example a total bandwidth maximum restriction may be applied to a whole cluster of peering partner's PSTN gateways modelled as a Realm. Also identical header-field manipulation and routing may be applied to all the machines in a Realm. Therefore the administrator needs to assign CAs to the Realms and associate the common rules with the Realm. The functionality of Realms' rules is the same as for CAs.

4.1.2 SBC Logic

It is important to plan what the SBC will actually do for your network in precise terms because particular features have further impact on capacity planning, integration with other components, interoperability and administration.

Routing

What will be the routing criteria used in your network?

Routing is a mandatory part of every SBC configuration. Once the topology is established, you must define how traffic flows between the Realms and Call Agents. That is described in routing tables. The key decision to be made is what is the criteria used to determine the next hop for a new session. The most common examples of criteria include:

- prefix-based routing. This is frequently used when you have a number of PSTN gateways serving different regions. Technically you match area codes against beginning of the user-part of the request URI.
- source-based routing. This is frequently used when you connect multiple IP networks and want to make sure that all traffic from one network is forwarded to the other and vice versa. The criteria is then the source IP address, source Call Agent or Realm.
- method-based routing. Sometimes specialised servers are used for processing specific traffic, like message stores for keeping messages for off-line recipients.
- The SBC configuration options include even more criteria and these can be also combined with each other.

Some functionality is only present in some deployments and whether to use it or not depends on used equipment, network characteristics and network policies. More information about administration of SIP routing may be found in the Section *SIP Routing*.

Media Anchoring

Do you need the SBC to anchor media so that all RTP traffic visits your site?

The ideal answer is no due to latency and bandwidth concerns, the most common answer is yes due to NAT traversal and controlling media. Relaying media costs considerable bandwidth and implies more SBC boxes. Yet if any of the following conditions applies, you will have to enable media relay:

- There are SIP clients behind the NATs. That's the common case in residential VoIP.
- You wish to record calls. Obviously you can only record media that visits the SBC.
- You want to implement topology hiding consequently and make sure that no party sees media coming from any other IP than that of the SBC.
- The SBC connects two networks that are mutually unroutable.

More administrative details can be found in the Section *Media Anchoring (RTP Relay)*.

Media Restrictions

Do media-restricting rules need to be placed?

The need for media restrictions arises mostly when bandwidth is scarce. This may be the case if media anchoring is used on a link from/to the SBC or on the SBC itself. It may be also the case on the link to the client, particularly if it is a mobile one.

The simplest solution is to restrict media negotiated by Call Agents by putting desirable codecs on a whitelist. All other codecs will be removed from codec negotiation. It may happen though that the resulting codec subset is empty and the Call Agents would not be able to communicate with each other.

If there are no codecs left, you may extend the codec set by transcoding. The SBC then adds additional codecs to the negotiation process and if the Call Agents choose it, the SBC will convert media to the chosen codec. The penalty that needs to be considered is degraded throughput of the SBC.

In addition to the codec-based proactive bandwidth saving approach, the SBC can also limit bandwidth retroactively and put bandwidth limits on CAs or Realms (or some portions of its traffic). This helps to stay on the bandwidth budget even if SIP devices exceed traffic signalled in SIP. However, it remains a reactive measure. That is, it does not prevent excessive traffic, it just drops it and impairs the affected media streams.

Codec handling is described in the Sections *Media Type Filtering*, *CODEC Filtering*, *CODEC Preference* and *Transcoding*, administration of media limits is described in Section *Traffic Limiting and Shaping*.

Registrar Cache

Does the SIP traffic include REGISTER messages?

If so, we recommend that you do enable registrar cache. The cache is optimised to reduce the REGISTER traffic that is passed down to the registrar. This is particularly important if the clients are behind NATs. Then the cache must be configured to force SIP clients to re-register every minute to stay connected from behind NATs. Also the ability to track registration status of users allows the SBC logic to differentiate call processing for online and offline users. This can be used for example for voicemail routing.

Further administrative details are described in the Section *Registration Caching and Handling*.

NAT Handling

Are there some CAs behind NATs?

If so, you not only have to anchor media as described above, but also make sure that the signalling protocol traverses NATs successfully. Also registrar-cache must be used to force clients to refresh their connectivity using frequent re-registrations. Some deployments with STUN-capable SIP phones also set up a STUN server to assist these phones.

NAT configuration is described in further details in Sections *NAT Traversal* and *Media Anchoring (RTP Relay)*.

SBC High Availability

Shall the SBC be operated in high-availability mode?

While this is normally the case, small enterprise deployments may prefer buying and administering fewer boxes. Introducing high-availability requires a standby spare machine for every active SBC and effectively doubles the number of machines.

It is recommended that in a high-available configuration setup an administrative network is used for internal inter-node communications and the availability protocol used between the machines in the active/standby pair.

More administrative details about HA mode are available in the Section *High Available (HA) Pair Mode*.

Downstream Failover and Load-Balancing

Shall the SBC seek alternative destinations when primary destinations become unavailable?

Handling downstream failover may or may not be needed. For example if the downstream telephones are single-user SIP telephones, there are usually no backup devices. Some high-density devices like PSTN gateways implement automated failover in a way which is invisible to the SBC and the SBC doesn't need to handle it either. However if the primary destinations have spare backup machines without automated failover, the SBC can still detect a failure and try the alternate destinations.

If there are multiple alternate destinations, it may be also practical to spread the load among them.

There are several ways how to define a set of alternate destinations and their priorities: it can be defined in DNS maps or in the Call Agent specification. If the definition is managed in DNS, the SBC resolves DNS names automatically in compliance with **RFC 3263**. If using DNS is not practical, the same effect can be achieved by associating multiple IP addresses with a Call Agent. Additionaly, a backup Call Agent may be also associated with a Call Agent: in that case traffic to the backup destination will be processed by additional C-rules specific to the destination.

Procedures for determining the next hop are described in Section *Determination of the IP destination and Next-hop Load-Balancing*.

Dialing Plan Mediation

Do different CAs and Realms connect to the SBC use different dialling plans?

Often SBCs connecting different sites that use different numbering conventions: short-dials, regional dialling plans, special services numbers. To enable interconnection of such sites and avoid number overlaps, the SBC must bring all the numbers to a common denominator, mostly the E.164 numbering format.

More about mediation can be found in the Section *SIP Mediation*.

4.1.3 Security Policies

Generally, the SBC has two ways for protecting networks: putting various restrictions on traffic and concealing network internals. The latter is sometimes a double-edged sword as obfuscation of SIP traffic makes it hard to troubleshoot.

Restricting Traffic from Unwanted Sources

How do you identify and discard illegitimate traffic?

There are several ways the SBC recognises and drops undesirable traffic.

At the SIP-level you may set a variety of criteria which if it is met results in declining a session request. The conditions may include:

- unusual message patterns such as User-Agents of a type known to offend other SIP devices, URIs to premium numbers or simply anything else that can be matched
- unusual traffic patterns, such as call excessive call rate, number of parallel calls, or RTP bandwidth consumptions
- The traffic patterns apply statically to a whole Realm or Call Agent. However they may be also tied dynamically to "traffic from any single IP coming from the Realm" or "traffic to any single phone number". This way you could for example impose a Realm condition "maximum one parallel call from a single IP address to a 900 phone number".

Additionally, if traffic from some specific IP address begins to take really excessive dimensions, you can drop it straight at the IP layer before it reaches the SBC logic.

More information about filtering unwanted traffic can be found in Section *Police: Devising Security Rules in the ABC SBC*.

Topology Hiding

Do you prefer SIP transparency across networks or concealing network information?

This is indeed an operational dilemma. If you process SIP traffic "by standards", the traffic will be passing the SBC with minimum changes. This approach will reveal lot of information about one network to the other: which IP addresses are being used, which port ranges, what type of equipment and potentially even more. This makes life easier for attackers seeking security holes in networks and therefore some operators chose to obfuscate this information.

The penalty for traffic obfuscation is significant however: operators' administrators will find it similarly hard to find out what's is going on in their own networks. That doesn't make troubleshooting easier. Some complicated applications in which SIP messages tend to refer to each other (such as in call transfer) may also fail.

The choice to obfuscate or not is eventually to be taken by the operator. The ABC SBC has the following means of doing that:

- Topology hiding rewrites known SIP header fields in which use of IP addresses is mandatory. The downside is that troubleshooting becomes more difficult.
- Use of non-transparent mode will rewrite dialog-identifying information: from-tag, to-tag and callid which in some older implementations also includes IP addresses. The downside is some applications which refer in protocol to a call may fail.
- Header whitelisting drops all header-fields that may potentially carry additional sensitive information, standardised (*Warning*, *User-Agent* for example) or proprietary (*Remote-Party-ID* for example). The downside is that sometimes "a baby can be thrown out with the bath water", when the header-fields include potentially useful information.
- Media anchoring can be used to obfuscate where media flows from and to. The downside is high bandwidth consumption and increased latency if media anchoring wouldn't be used otherwise.

Additional information can be found in the Sections *Topology Hiding*, *SIP Header Processing* and *Media Anchoring (RTP Relay)*.

4.1.4 Capacity planning

Capacity planning is a key part of the planning exercise. Failure to provision resources sufficiently can lead to network congestion and low quality of services. Overprovisioning way too far increases cost. The goal is to find the right measure of network size that serves the anticipated traffic. This section provides rules of thumb to estimate needed capacity and makes simplifying assumptions about state-of-the-art hardware, "normal" traffic patterns and no dependencies on external servers. A more detailed discussion of dimensioning can be found in the Section *SBC Dimensioning and Performance Tuning*.

Cluster Size

How many SBCs are required for a deployment?

There are two major factors that determine how many hosts you need to serve your traffic: anticipated performance bottleneck and organisation of clusters.

Which bottleneck is the most critical strongly varies with actual traffic patterns and services configured on the SBC. A rule of thumb for a rough estimate of the performance of the ABC SBC on PC with three-1GB-Ethernet and 12 GB of memory is this:

- If transcoding is used, the bottleneck of a single box in terms of the maximum number of parallel calls which is about 1000. Otherwise...

- ... if media-anchoring is used, the bottleneck in terms of the the maximum number of parallel calls which is about 5000. (Media overhead prevails even over heavy registration load.)
- Otherwise the limit is a call rate of 480 calls per second.

We advice to add at least additional 35% of buffer capacity to deal with variances in hardware performance, increasing traffic patterns, too conservative traffic forecasts and DoS attacks.

Once you determined the per-box capacity, you need to take cluster organisation in account. There are the following three cases:

- a single SBC deployment: no scaling, no high-availability.
- high-available active/standby pair: the pair has still the total capacity of a single box, however it can survive scheduled and unscheduled outages without service impairment
- high-available cluster: the number of boxes is determined by number of boxes needed to serve the target capacity, doubled to achieve high availability plus two more boxes for a highly-available load-balancer: cluster_capacity / box_capacity * 2 + 2.

Bandwidth

How much bandwidth needs to be allocated to serve the deployment?

To determine needed bandwidth you need to discriminate between two cases: using SBC with and without media anchoring. The more bandwidth-hungry case is that with media anchoring. With the most commonly used codec, G.711, a call consumes 197 kbps bandwidth in each direction.

To determine the maximum bandwidth needed calculate the product of maximum number of parallel calls by the bandwidth specific to the codec in use, 197 kbps if it is G.711.

Public IP Address Space

How many public IP addresses need to be allocated for an SBC Cluster?

The minimum number is one shared VIP address for every active/standby pair.

4.1.5 IT Integration

An SBC is rarely a standalone component. More often it integrates with other components for the sake of connecting to external policy logic, network monitoring, server naming and others. This section lists typical integration options you may need to consider for your deployment.

RESTful interface

Does the SBC need to consult an external server for its decision making?

If so, the ABC SBC built-in RESTful query allows to ask an external server how a session shall be handled. This query possibility allows to integrate external and complicated logic in the SBC which is customer-specific or for other reasons difficult to integrate with the SBC directly.

See more in the Section *RESTful Interface*.

Recording

For various reasons, audio recording may need to be configured. What needs to be integrated is access to the recorded files. The easiest way is none: the recorded files are stored on local storage and accessed through the events web-page. Uploading to HTTP may also be used. In either case, some deletion and retention policy must be created, otherwise the local storage will be soon full.

See more in the Section *Audio Recording*.

Monitoring

Do you need to see how the SBC is doing?

Of course you do. We suggest use of the optional ABC Monitor as described in Section *ABC Monitor (Optional)* as it provides ABC SBC administrators with full history of users and analytical tools to audit it.

You can also use SNMP at a third-party management console to inspect health of your ABC SBC and the networks. It is possible to define your own custom counters. See more in the Section *Using SNMP for Measurements and Monitoring*.

Mass Provisioning

Do you need to provision the SBC with lot of repetitive data such as thousands of Least-Cost-Routing Entries?

Then you certainly do not want to provision it rule by rule. Instead you devise one rule and fill it with data. The actual data can comes over a web interface or RPC.

See more in the Section *Provisioned Tables*.

Call Detail Record (CDR) Exports

Do you need to access CDRs for sake of charging and reconcilliation?

Then you must access the internally produced CDRs.

See the Section *Call Data Records (CDRs)* for more about CDR location, format and access.

DNS Naming

How do I make the SBCs known to their counter-parts?

While it is possibly to communicate with peer SIP-devices only using IP addresses we recommend that every single SBC has a DNS name which is communicated as the point-of-contact to its peers. If nothing else, it makes IP renumbering much easier should it occur.

DNS map entries for SIP servers follow the SRV DNS extension as described in **RFC 3263**.

4.2 Planning Checklists

This section provides you with a summary of questions raised in the previous section. We urge that you dilligently check all the items before you proceed with commencing an installation.

Topology

- Have you identified all Call Agents present in your network?
- Have you specified additional processing rules for these Call Agents, such as network limits?
- Have you grouped all Call Agents in Realms present in your network?
- Have you specified additional processing rules for these Call Agents such as network limits?

- Have you specified all physical interfaces (Ethernet cards)?
- Have you specified all IP addresses, port ranges, and VLANs to be used on these interfaces?
- If there are firewalls in front of the SBC, have you verified that all needed ports are open?
- Have you verified that IP rules on the SBC restrict traffic to privileged ports from trusted IP addresses only?

SBC Logic

- Have you devised the SIP routing criteria used in your network? How many routing rules do you anticipate?
- Have you devised the routing flows between Realms and CAs?
- Does any of the conditions mentioned necessitate use of media relays?
- If you need to restrict codecs, which codecs shall be permitted and which codecs shall be restricted?
- Do you need to force use of a codec unsupported by a CA using transcoding? Which codec?
- If your SIP traffic includes REGISTER messages, will you enable registrar cache? If so, what will be the registration interval?
- Does presence of clients behind NATs necessitate use of media-relay, symmetric SIP and registrar-cache?
- Do you plan to set up high-available SBC pair(s)?
- If you use the high-available (HA) SBC pair, do you plan to use an administrative network for the HA protocol?
- If you need to handle downstream failover, have you devised appropriate DNS maps?
- Does the SBC accept traffic using different dialling conventions? If so, how will you translate between them?

Security

- What conditions do you devise to drop illegitimate traffic? Will you configure IP-based and/or URI-based blacklists?
- Will you introduce traffic shaping limits: call-rate, call-length, parallel calls and maximum call-length?
- Will all or only registered SIP devices be permitted to make phone calls?
- Will the need to troubleshoot your network easily or the need to hide topology prevail?

Dimensioning

- How many SBCs do you need?
- How many network cards shall each SBC have?
- How many IP addresses do you need?
- How much bandwidth do you need in each direction?

Integration

- Do you plan to use the management components over a dedicated administrative network?
- Is external session decision-making logic using RESTful interface needed? If so, what are the parameters passed from and to the RESTful server?
- Is SNMP monitoring needed? If so, what is the SNMP configuration data (IP address, SNMP community)?
- Do you need to mass-provision some configuration data? What is the structure of the data and what size of tables do you anticipate?
- Do you need to record audio and access it? What is your deletion and retention policy for the stored audio files?
- Do you need to export CDRs?
- Have you devised appropriate DNS SRV and A entries for all IP addresses?

4.3 A Typical SBC Configuration Example

Many SBC deployments, especially in smaller networks, follow a simple schema which is given through the network structure. In this typical network, the SBC bridges between an internal network, where the home proxies, PBXs and other servers like conference and application servers are located, and the public network, where the user agents reside. Typically, in such a network the main motivations for deploying an SBC are

- network separation for security reasons
- foolproof and always-working NAT handling
- protection of the core network from high registration load
- protection against fraud by enforcing call limits
- possibility for monitoring and tracing for troubleshooting

This chapter presents step by step how to address these network aspects using the ABC SBC. It assumes an SBC "sitting" between two networks, a public one with user telephones and a private protected one with operator's infrastructure.

4.3.1 Identifying Network topology

Simple as it is in this case, the network topology is shown in *Sample network Topology*.

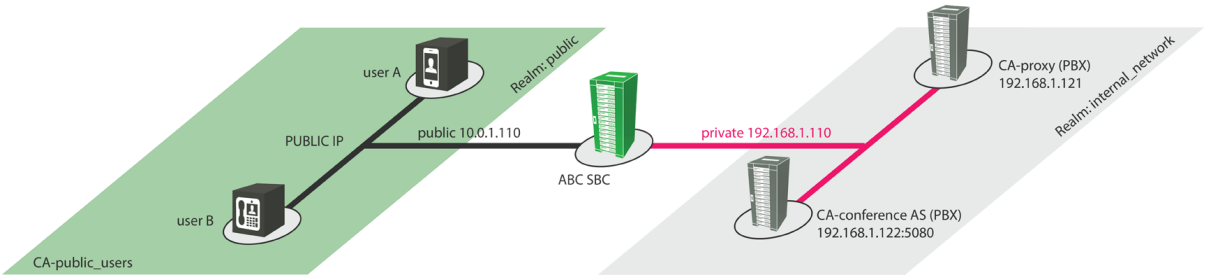


Fig. 1: Sample network Topology

What administrator needs to do in this step is configuration of the physical network interfaces and of the SBC-level interfaces.

The ABC SBC has two physical interfaces, one *public* connecting to the public networks, here with IP address 10.0.1.110, and one *private* connecting to the private network, here with IP address 192.168.1.110. The physical interfaces are configured using procedures described in Section *Physical and System Interfaces*.

User agents are located in the public network and have IP addresses from any network, and they are configured to use the public interface of the SBC with the address 10.0.1.110 as proxy (in a real world deployment, this address would not be a private RFC1918 address, but a public one).

A proxy (or PBX) and a conference (or other application) server are located in the internal network. The ABC SBC can communicate with the entities in the internal network through its interface in the private network which has the IP address 192.168.1.110.

The detailed procedure for setting up SBC interfaces is described in Section *SBC Interfaces*. It links media processing, signaling and administration with physical interfaces, IP addresses and port ranges.

4.3.2 Describing ABC SBC Realms and Call Agents

The network topology is described in the ABC SBC configuration by Realms and Call Agents. Call Agents are typically consumer or operator SIP devices identified by their IP addresses or DNS names. They are grouped in networks called Realms whose processing rules they share.

In our example two Realms are created in the SBC: *public* and *internal_network*.

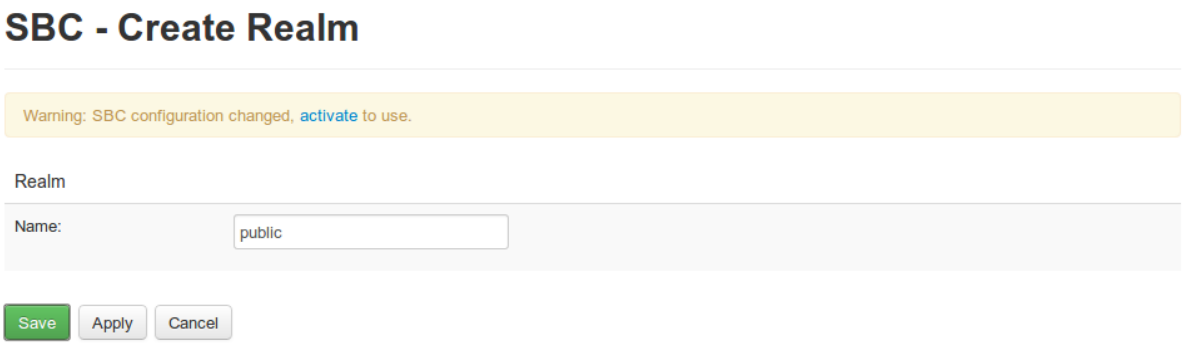


Fig. 2: Creation of Realm

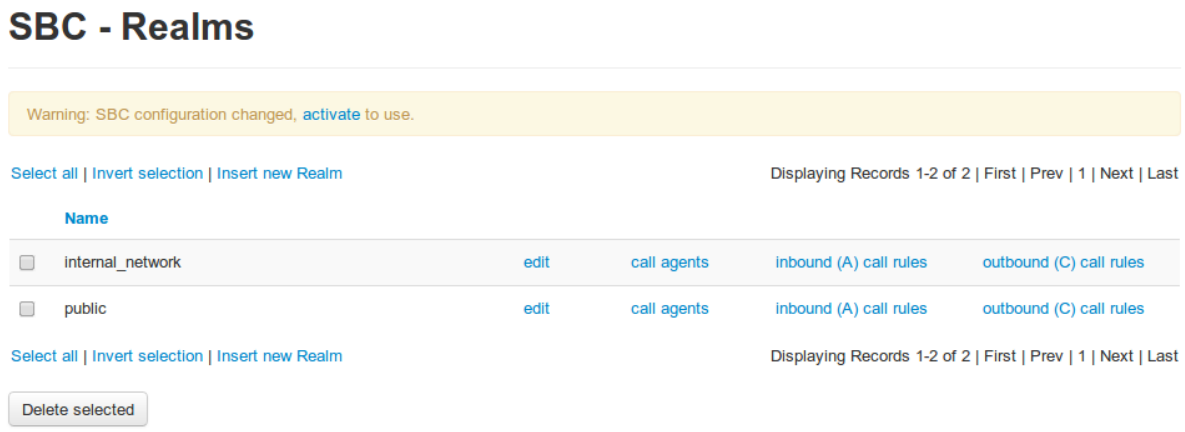


Fig. 3: Public and private Realms

In the Realm *public*, the call agent *public_users* is created with IP address 0.0.0.0/0, which means that public_users can have any IP address, or: requests received from any IP address on the public interface will be identified as coming from the Call Agent *public_users*. The address list can include multiple addresses that are used for routing (See section *Determination of the IP destination and Next-hop Load-Balancing*). Also a backup call agent can be defined here which can be used as alternate destination if forwarding to the primary destination fails. The CA definition further specifies interfaces used for sending and receiving signaling and media and availability management information – see Section *IP Blacklisting: Adaptive Availability Management* for more information.

The call agents could be assigned to SBC nodes and/or config groups. This assignment basically specify what SBC nodes is the call agent known to.

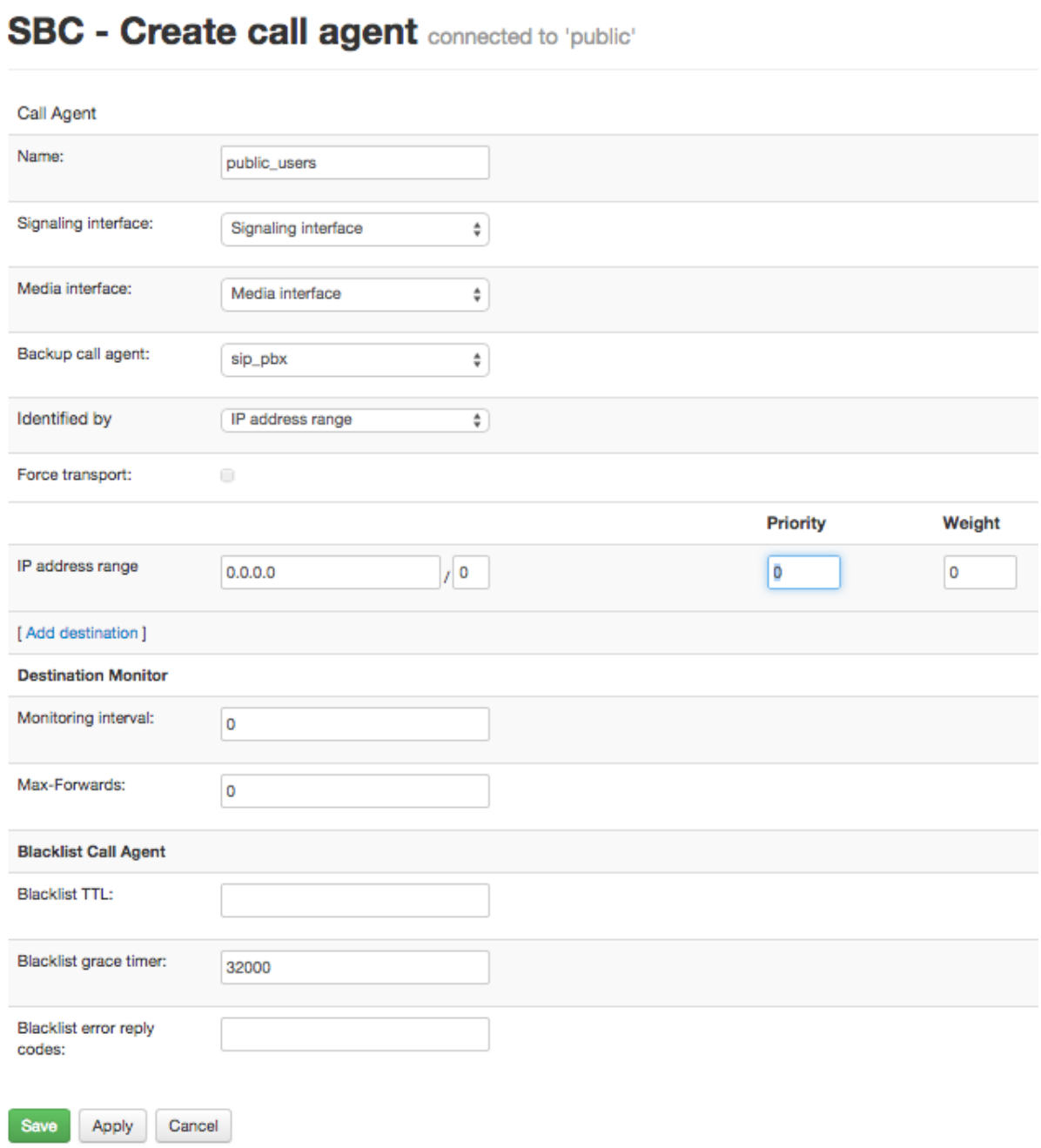


Fig. 4: Create public-users Call Agent

As we have neither defined a specific IP:port for the Call Agent nor a hostname, requests can be routed to that Call Agent only by Request URI, or by setting the destination IP explicitly in the routing rule.

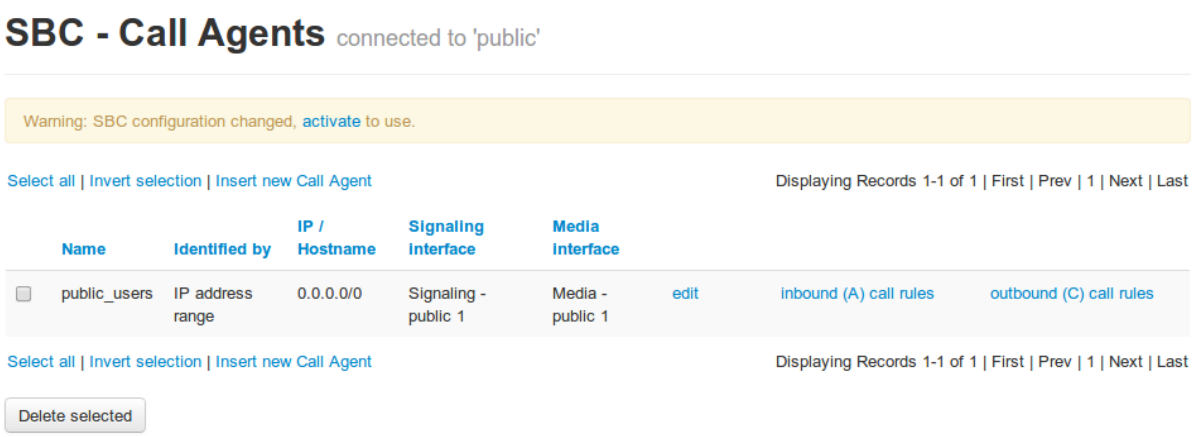


Fig. 5: Public Call Agents list

For the internal realm, the call agents *proxy* and *conference* are created with IP addresses 192.168.1.121 and 192.168.1.122:5080 respectively.

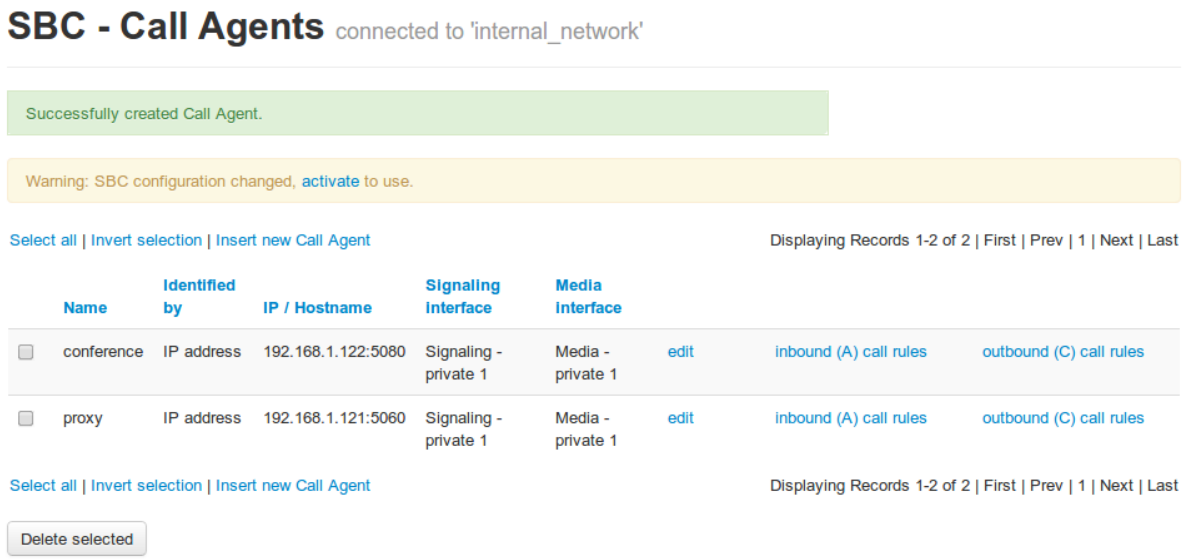


Fig. 6: Internal Call Agents list

Provisioning Call Agents Using RPC

It is also possible to provision Call Agents using XML-RPC interface.

The following RPC commands for Call Agent provisioning are available:

- cagents.fetch() - fetch all the Call Agents
- cagents.insert($payload) - insert Call Agent
- cagents.update($payload) - update Call Agent
- cagents.delete($realm_name, $cagent_name) - delete Call Agent

The $payload parameter of the *insert* and *update* functions is structure of following format:

```
{
    "realm": "Some_Realm_Name",
```

(continues on next page)

(continued from previous page)

```
    "name": "Some_Call_Agent_Name",
    "interface": "public_signaling",
    "media_interface": "public_media",
    "target": SEE_BELLOW
    "transport": "UDP",
    "backup_ca": {
        "ca":     "Backup_Call_Agent_Name",
        "realm":  "Name_of_Realm_Backup_Call_Agent_Belongs_To"
    },
    "backup2_ca": {
        "ca":     "2nd_Backup_Call_Agent_Name",
        "realm":  "Name_of_Realm_2nd_Backup_Call_Agent_Belongs_To"
    },
    "config_groups": ["Config_Group_Name"]
    "attrs": {
        "name_of_attribute_1": "value_of_attribute_1",
        "name_of_attribute_2": "value_of_attribute_2"
    }
}
```

For call agents identified by subnet, the target should look like this:

```
"target": {
    "subnet": ["0.0.0.0/32"]
}
```

For call agents identified by IP address or DNS name, the target should look like this:

```
"target": {
    "hosts": [
        { "addr": "192.168.1.1:5080", "weight": 10, "priority": 10 },
        { "addr": "192.168.1.2:5080", "weight": 10, "priority": 20 }
    ]
}
```

When updating Call Agent only the *realm* and *name* fields are mandatory, They identify the Call Agent to be updated. If any of the other fields is not specified it is not changed by the update action.

4.3.3 Configuring Registration Cache and Throttling

REGISTER processing accommodates several goals: off-loading servers behind the SBC, enforcing frequent re-registration load to keep NAT bindings alive and dealing with REGISTER avalanches caused by different sorts of outages.

For REGISTER requests coming from the "public side", the ABC SBC is configured to cache the registrations using the **Enable REGISTER caching** action. The cache works as follows:

- For every new registration, it creates an *alias*, a special unique one-time identifier.
- It saves the original contact along with the alias in the local registrar cache.
- To facilitate NAT traversal, it also saves the IP address, port and transport with which the REGISTER was received.
- It may re-adjust re-registration period so that it is frequent towards client for NAT keep-alives and less frequent downstream for better performance.
- It replaces the Contact in the REGISTER with a combination of the alias and the SBCs IP address: alias@SBC_IP:SBC_PORT.

This way, the "aliased" contact propagated downstream hides details of NAT-related address translation performed at the SBC and manipulates re-registration period as needed. The cache entry becomes effective once the REGISTER request is positively confirmed by the downstream SIP element.

Thus, when the REGISTER request is then routed to the registrar (the home proxy, here Call Agent *proxy*), the alias@SBC_IP:SBC_PORT is saved as he registered contact address of the user at the registrar.

We define this rule in the A rules of the *public* Realm, so that it is executed for REGISTER requests coming from any user agent defined under the Realm.

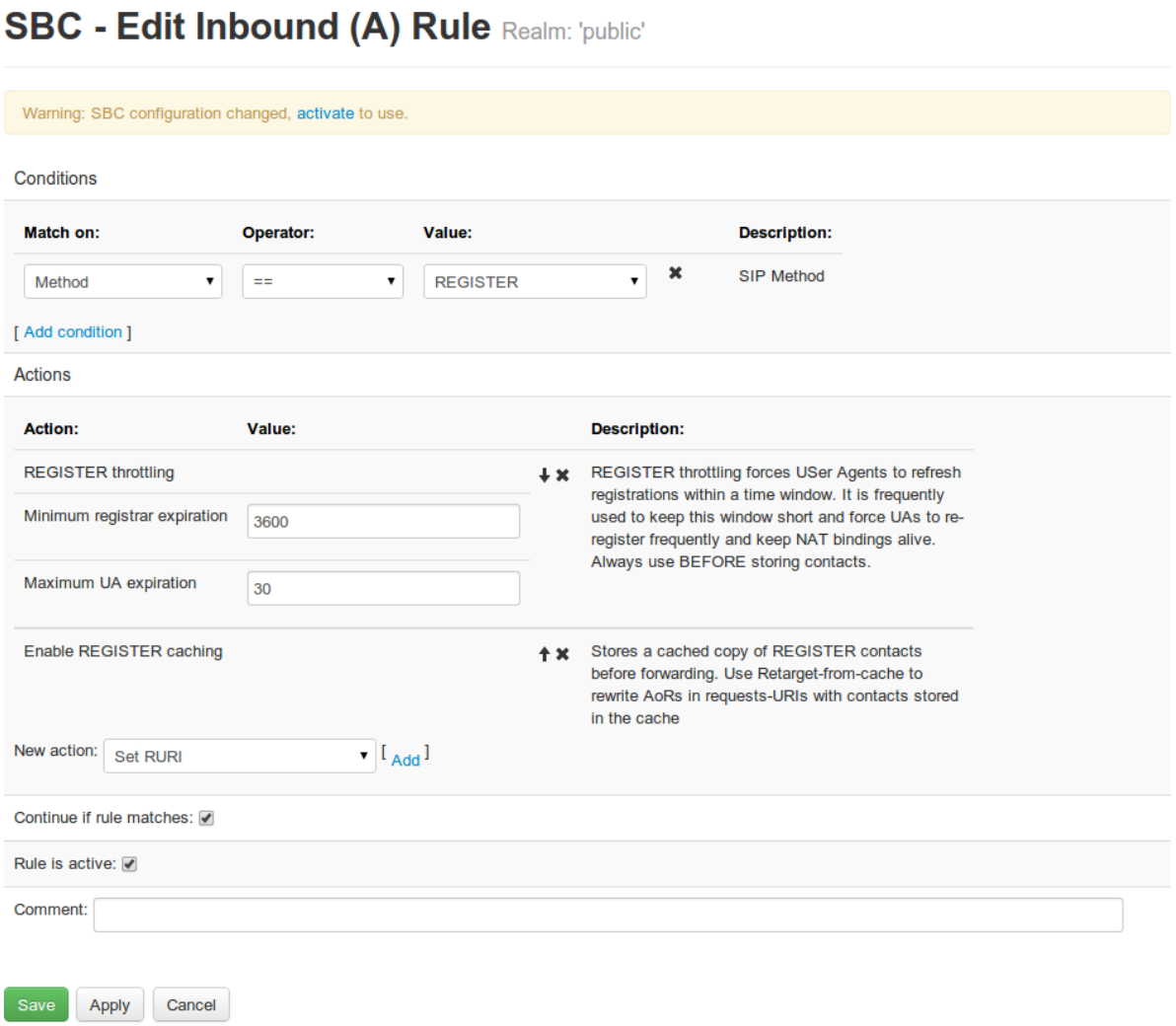


Fig. 7: Rule *A* Definition for caching REGISTERs coming from *public* realm

In order to protect the home proxy from the bulk of the registration load, the action **REGISTER throttling** is enabled with a **Minimum registrar expiration**, i.e., the re-register interval used upstream to the home proxy, set to the default of 3600 (one hour), while the **Maximum UA expiration**, i.e., the re-register period for the user agents, is set to 30 seconds.

4.3.4 SIP Routing

The SIP routing tables (B tables) define to which Call Agent a call is forwarded. In our example, there are two cases: calls from the UAs towards the proxy server and calls from the internal network towards the UAs.

Calls from the User Agents are routed towards the proxy with a simple rule. Here we route all calls from the public realm to the proxy - we might also set a filter on Source Call Agent, which would be equivalent in our case. We route by setting the next_hop (the destination IP address) directly.

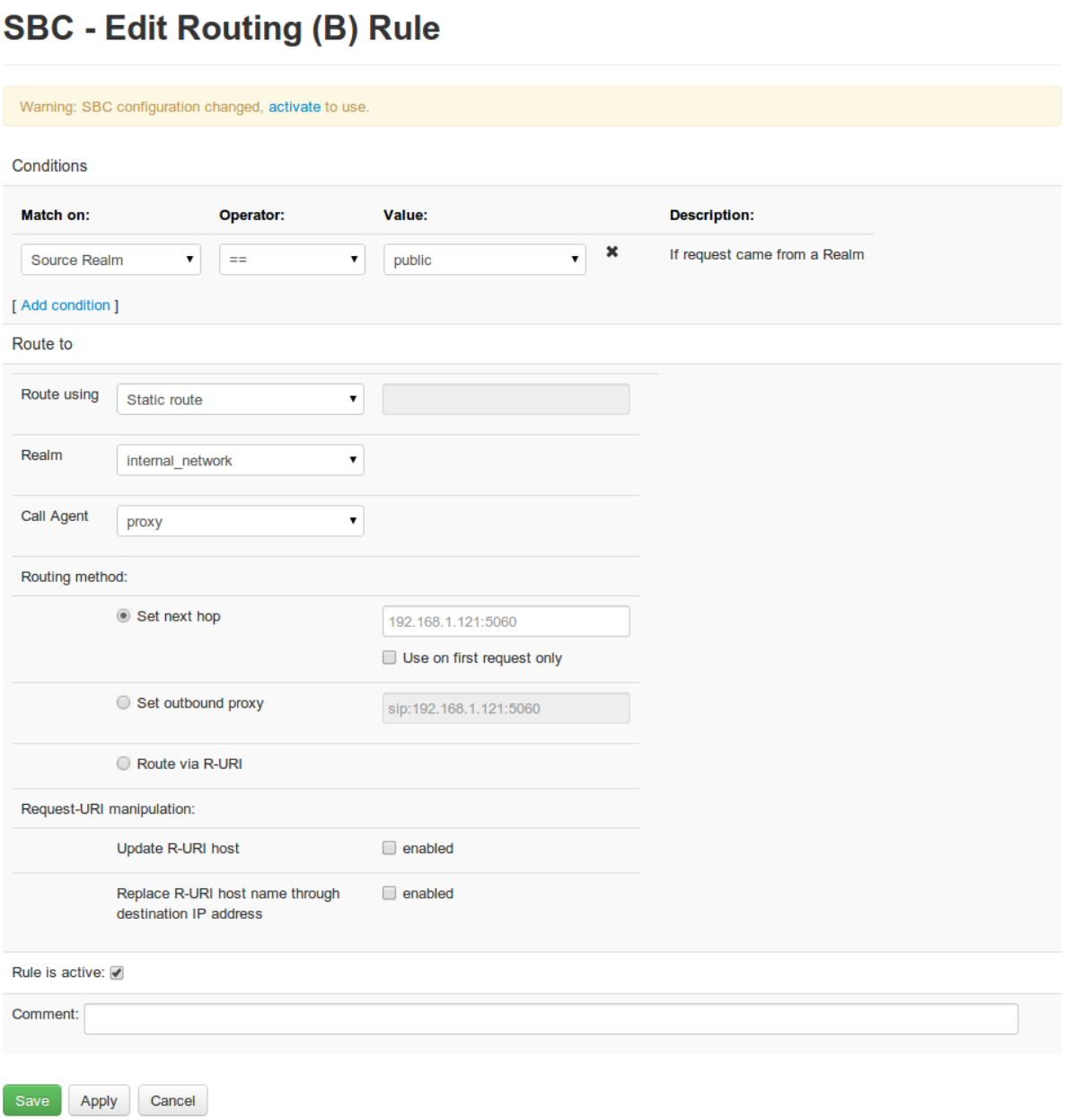


Fig. 8: Rule *B* Definition for the sample network

The next rule specifies routing of all calls from the internal network towards the registered UAs. If the home proxy wants to send a call to a user, it finds in its registrar database the alias@SBC_IP:SBC_PORT as contact for the user, thus it sends the call to the SBC with the alias in the request URI like this: INVITE sip:alias@SBC_IP:SBC_PORT.

In the SBC, we use the action **Retarget R-URI from cache (alias)** to look up the UAs IP and port values and set the request-URI to it. We also use the **Enable NAT handling** and **Enable sticky transport** options to handle

NATs properly. Using these options the SBC will send the request to the IP and port where the REGISTER request was received from and using the same transport protocol it was received on.

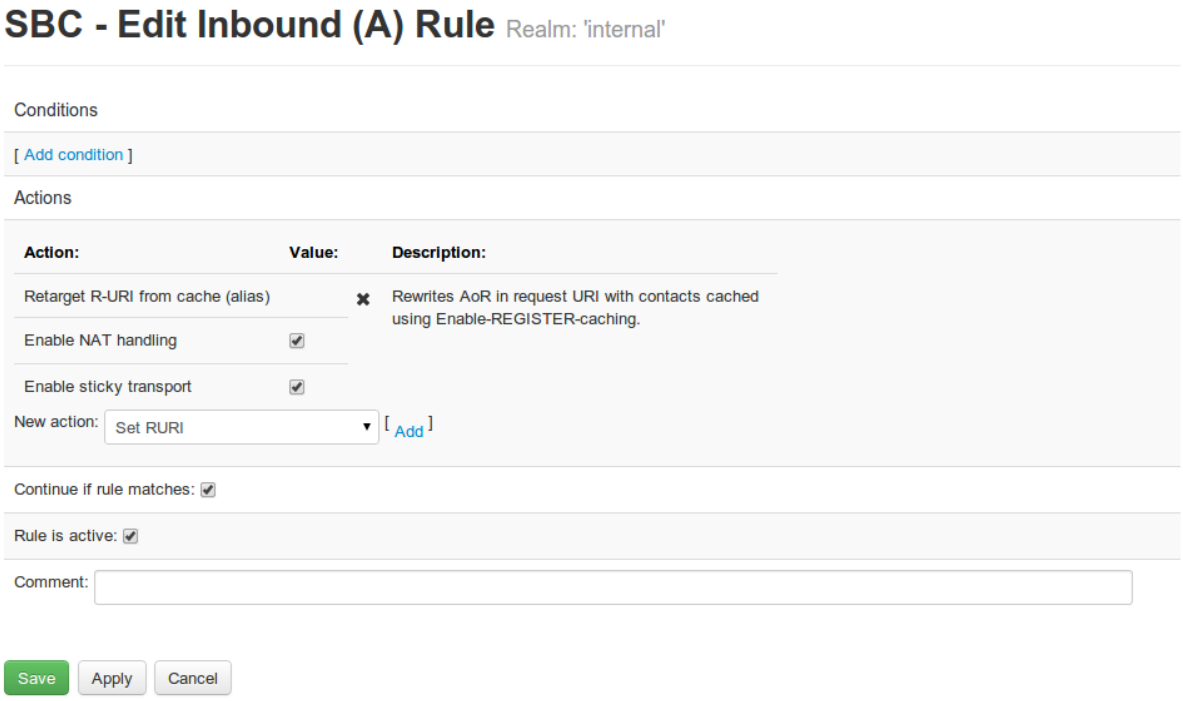


Fig. 9: Rule *A* Definition for *internal* CAs

We can then use the R-URI to determine request's destination. For simplicity, in this example we define a catch-all routing rule for the complete internal network, which includes all call agents defined there. (We may also define special routing rules for the different call agents in the internal network if they would have to be treated separately, e.g. if some calls need to be sent to a peering partner.)

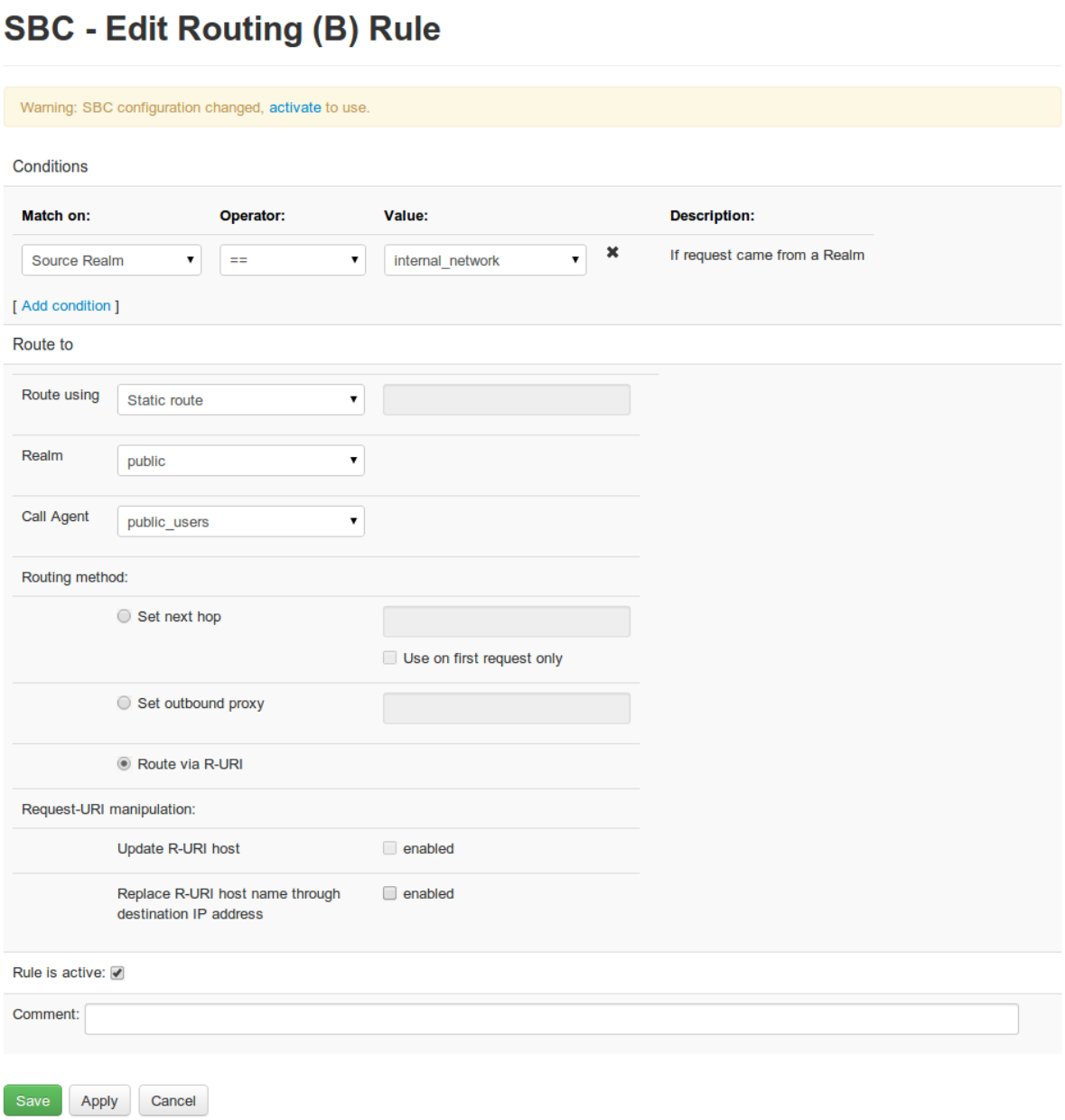


Fig. 10: Rule *B* Definition for *internal-network* Realm

4.3.5 Configuring NAT Handling and Media Anchoring

We have already used the NAT option in the **Retarget R-URI from cache (alias)** action above. In order to route in-dialog requests to the caller properly even if the UA is behind NAT, we use the **Enable dialog NAT handling** action. This will make the SBC remember the source address of the caller for the dialog and use that to send in-dialog requests.

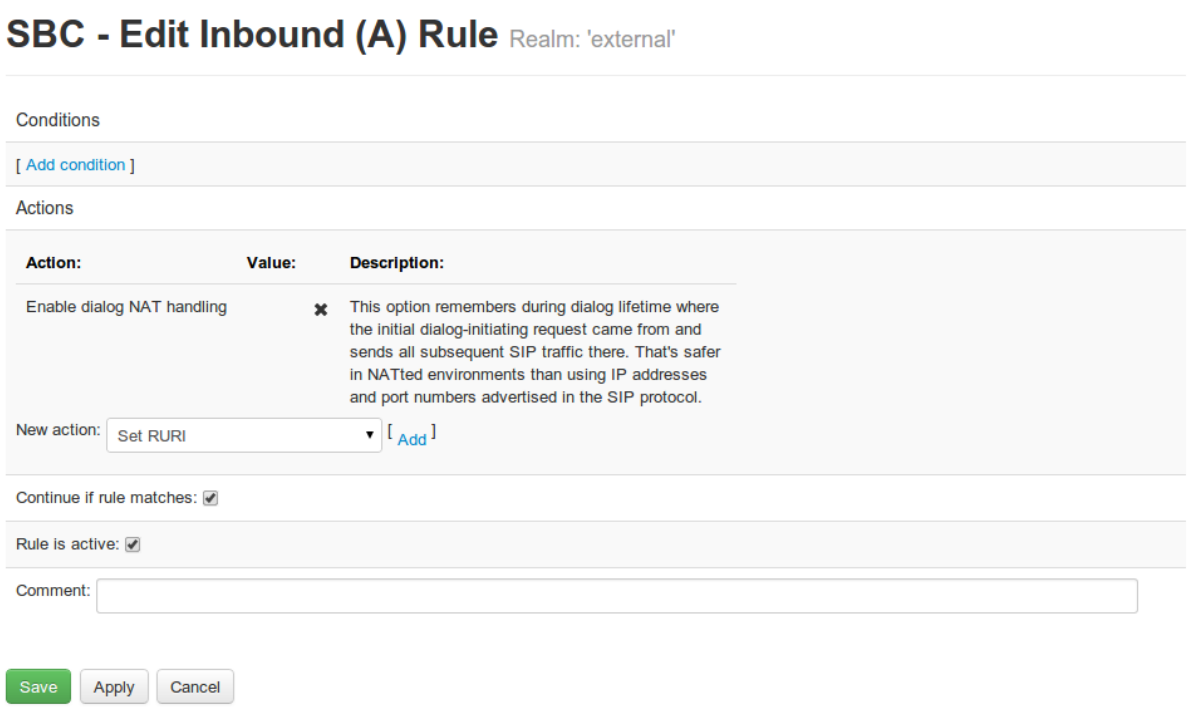


Fig. 11: Rule *A* Definition for NAT handling

For the RTP to flow properly through different NATed users - and also from the internal network to the public network for calls to conference bridge server - we **Enable RTP anchoring** with the **Force symmetric RTP for UAC** option enabled. To anchor the RTP of all calls at the SBC, we leave the **Enable intelligent relay** option unchecked; if we want to reduce bandwidth consumption and latency (total mouth-to-ear delay), we can also enable the intelligent relay option if we are sure that no users are behind double NATs.

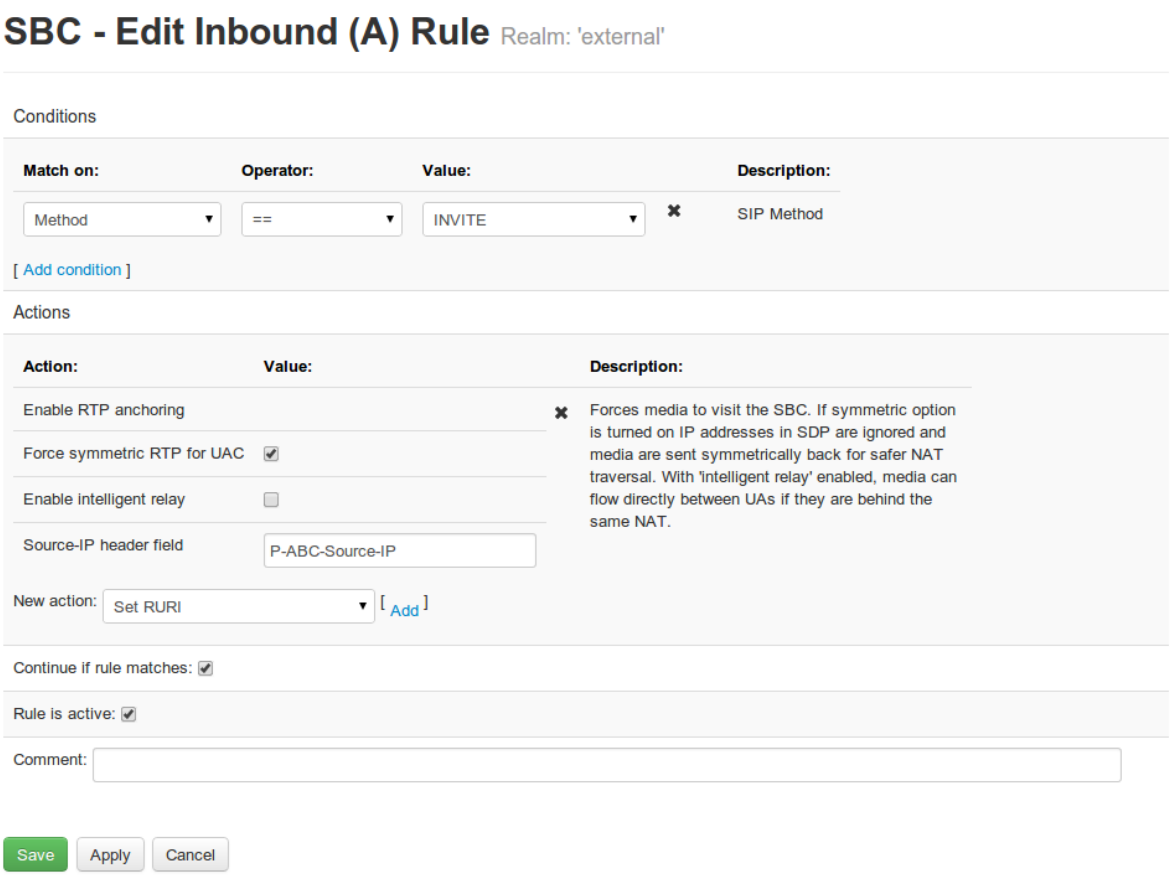


Fig. 12: RTP Anchoring Rule Definition (*A*)

We enable this for calls in both directions - from and to the UAs.

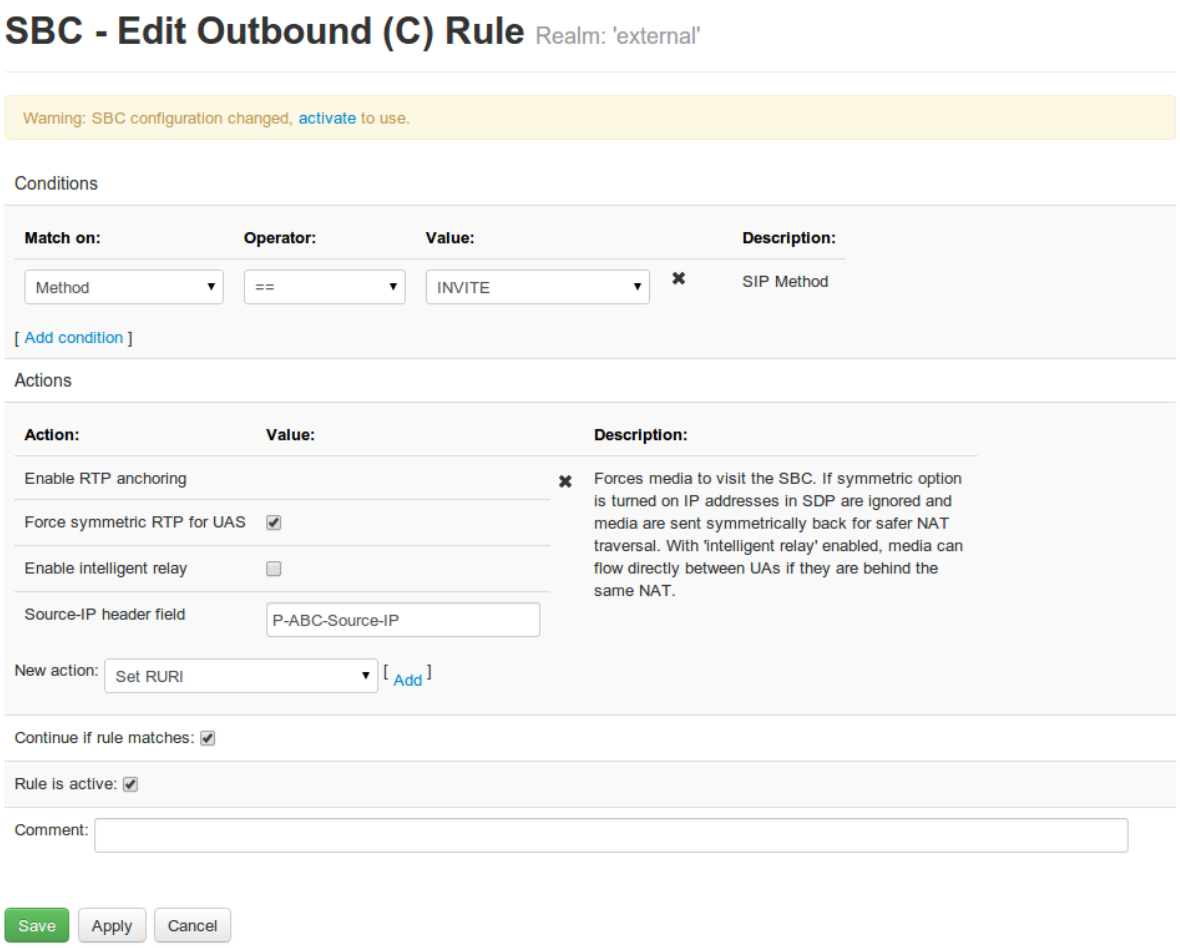


Fig. 13: RTP Anchoring Rule Definition (*C*)

4.3.6 Configuring transparent dialog IDs

If we want to enable call transfers through the SBC, and to simplify troubleshooting, we can **Enable transparent dialog IDs**.

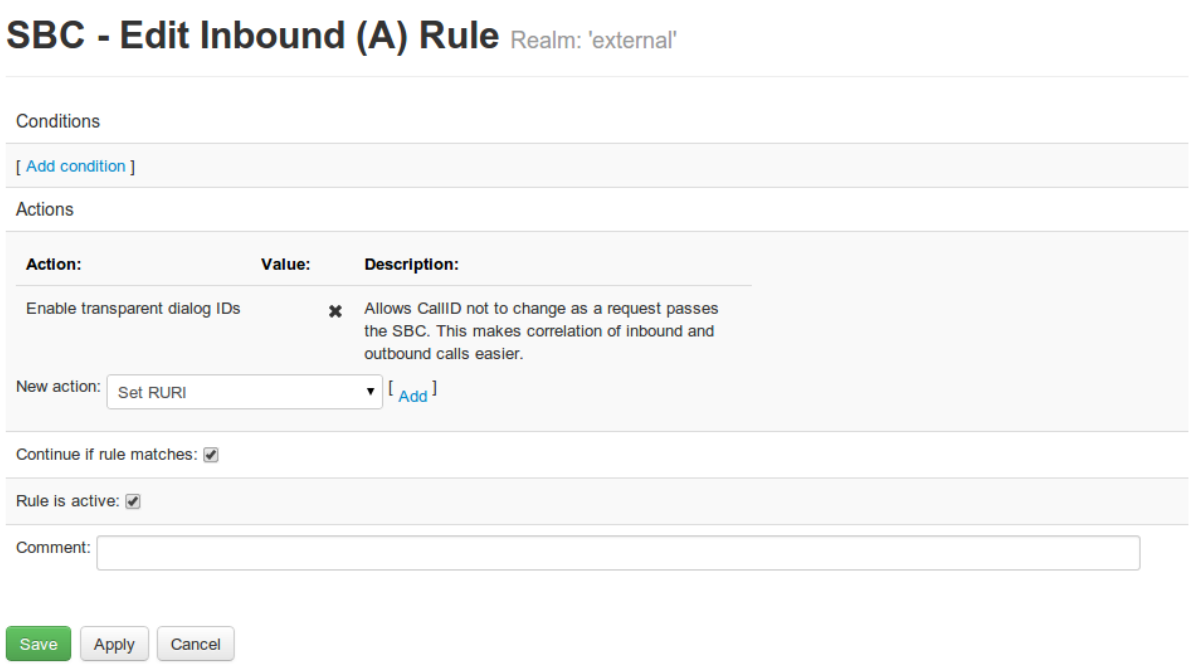


Fig. 14: Transparent Dialog Rule Definition (*A*)

4.3.7 Setting up tracing

In the testing phase, we can enable tracing for calls with the **Log received traffic** action.

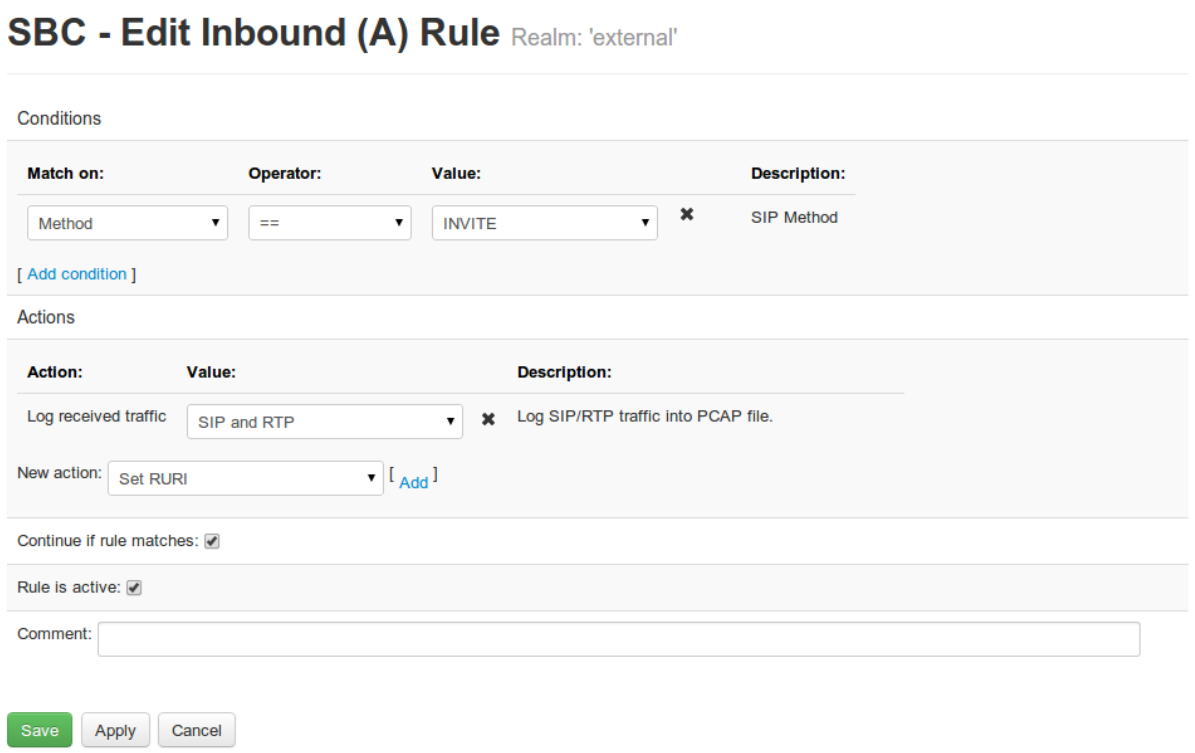


Fig. 15: Tracing Rule Definition (*A*)

In production use we should not forget to disable or remove this rule to protect the privacy of the users and to reduce processing power and disk space requirements at the SBC host.

4.3.8 Summary of rules

The rules we have created so far can be seen in the Overview screen. The rules implement so far routing from the external to the private network and vice versa, recording traffic in PCAP files, NAT handling and registration caching and throttling.

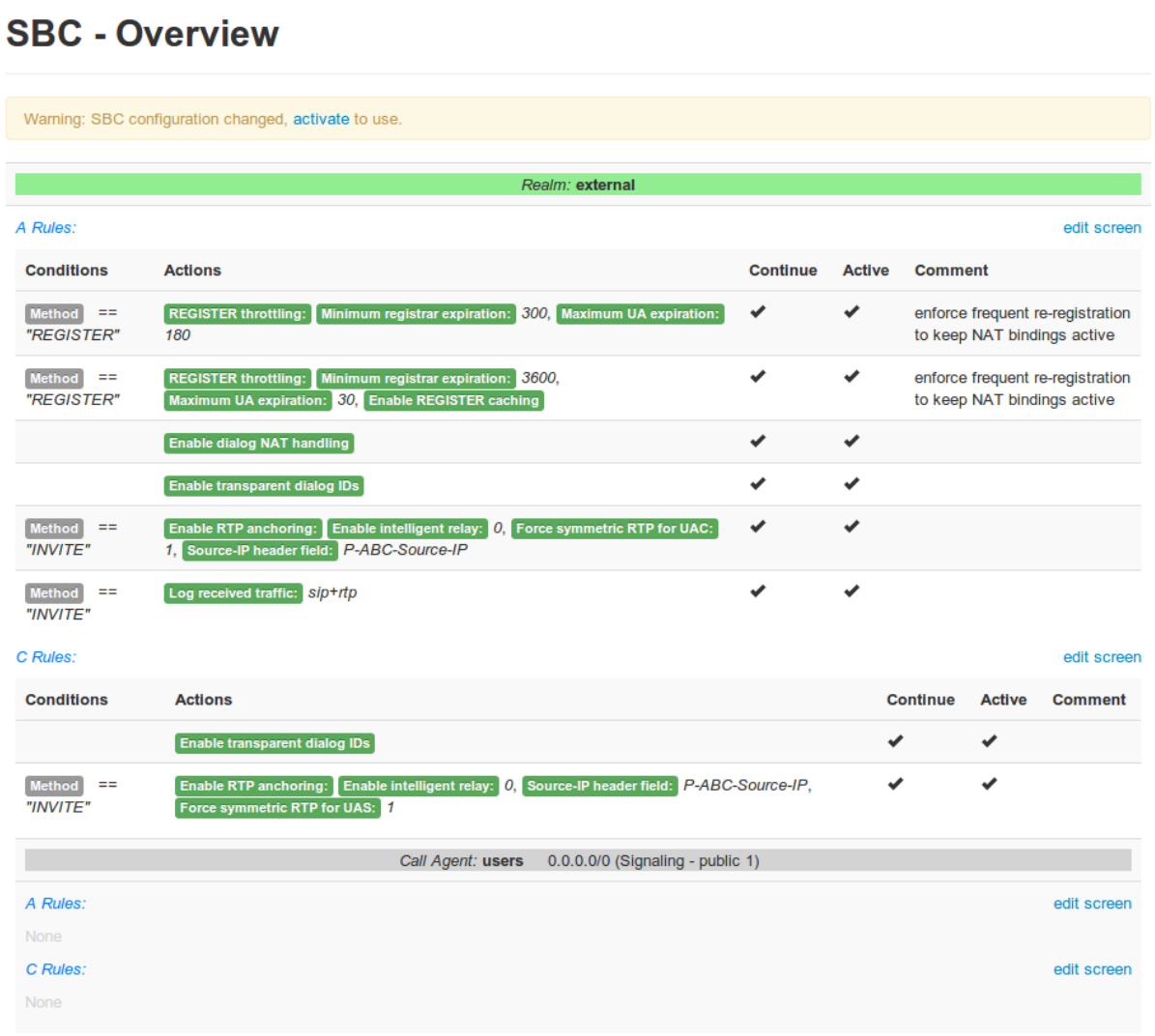


Fig. 16: Rule list for sample network

4.3.9 Setting Call Limits

In order to reduce the risks of fraud, we can set some limits on traffic coming from the external network as shown in Figure *Limiting calls and traffic*:

- a parallel call limit of 10 for calls coming to the realm from the same source IP address (**$si**)
- a limit of 5 calls coming to the realm from the same user (**$fU**)
- a limit of call attempts per second (CAPS) of 10 for the calls coming to the realm from the same source IP address (**$si**)
- and a limit of 120 kbit/s for every single call coming to the realm - sufficient bandwidth for audio calls only. For video calls you might want to use a higher value.

For the limits per source IP address, it has to be noted that the limits may apply to a group of users if they are behind the same NAT. If for example there are enterprise users, we may group them into a separate Realm with a

different, higher limit and/or group by a combination of IP address and domain name.

We set these limits in calls from the external realm.

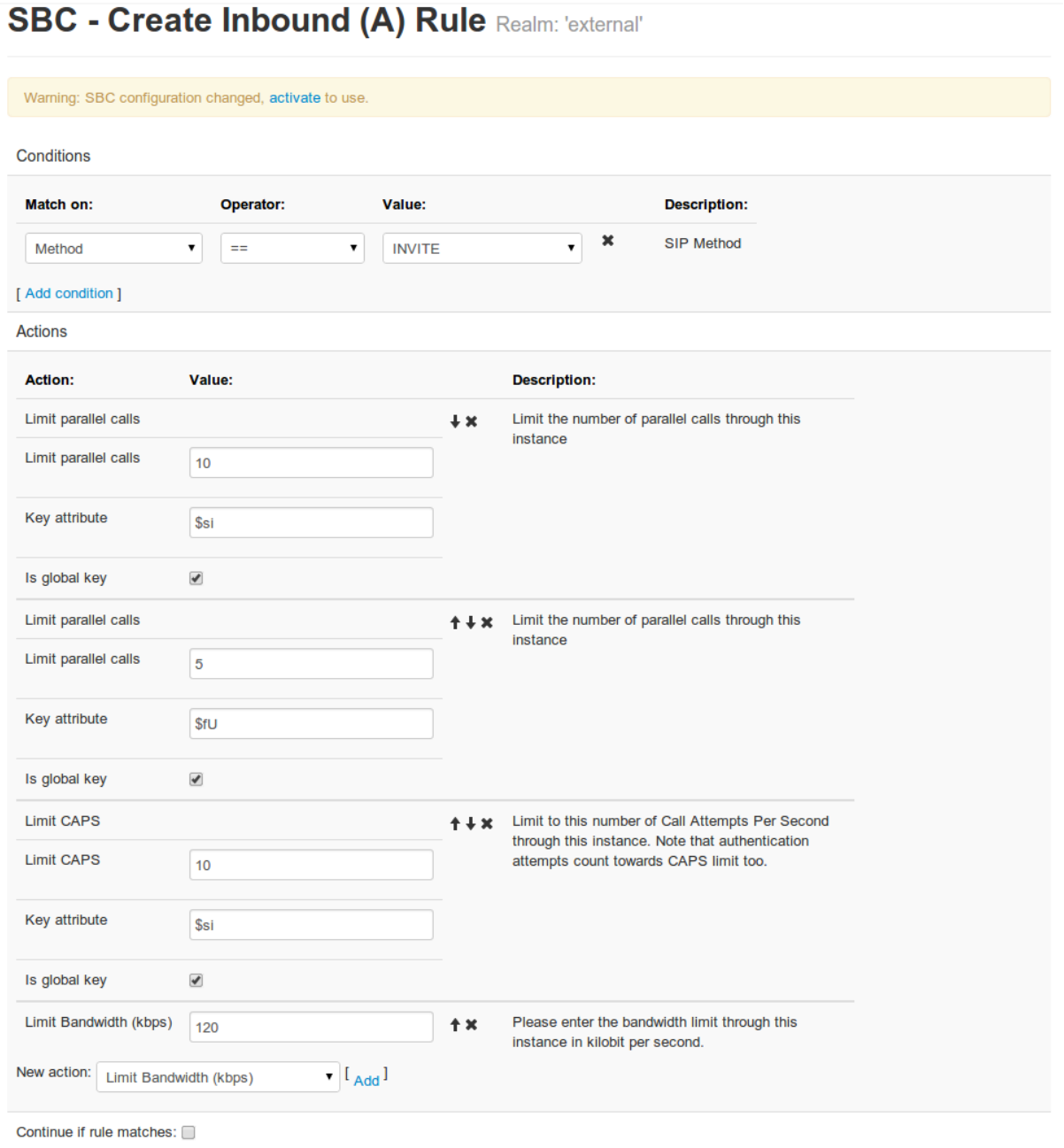


Fig. 17: Limiting calls and traffic

4.3.10 Blacklisting specific IPs and User Agents

We can use a rule to block calls from a specific IP address.

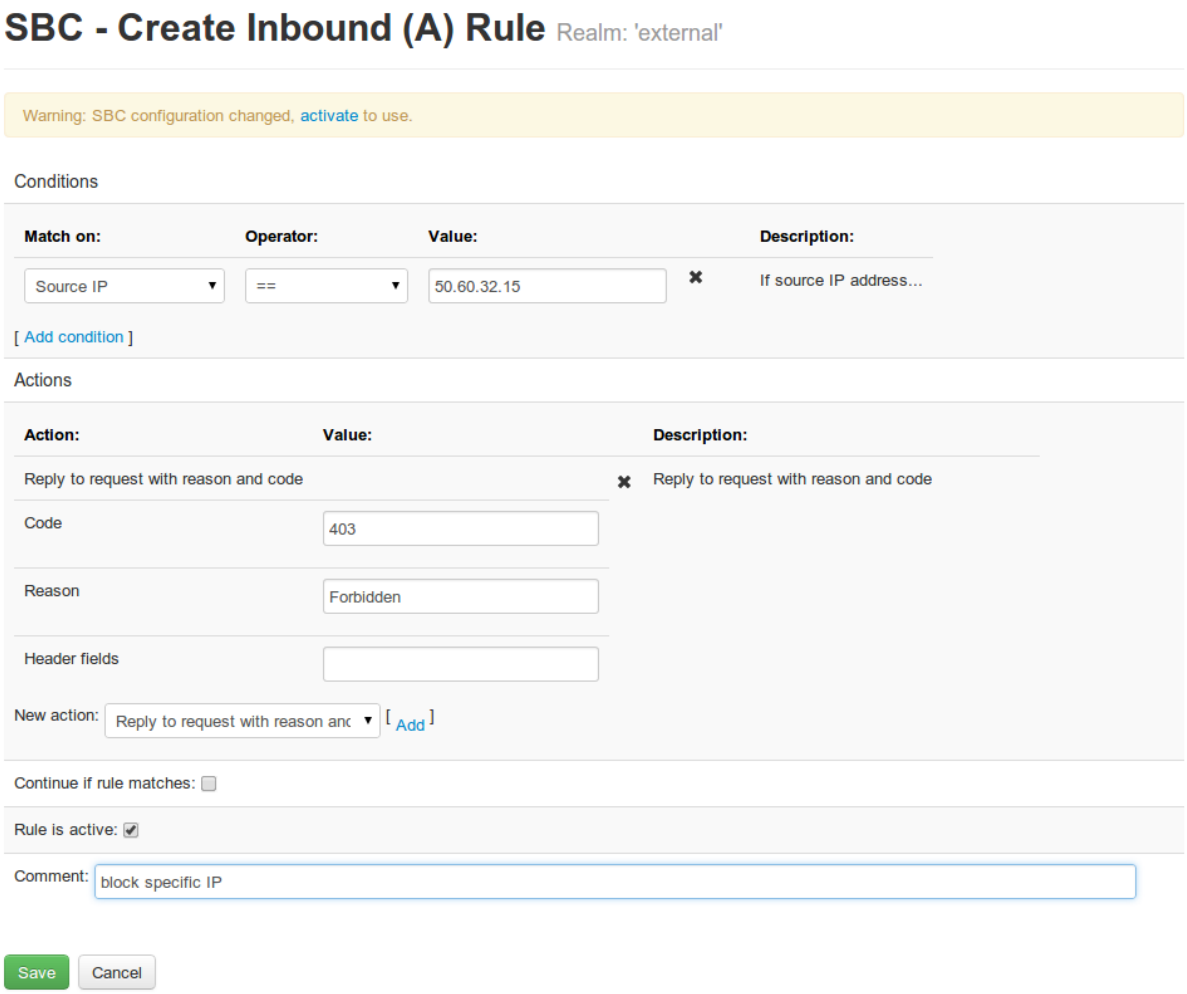


Fig. 18: Blacklisting IP addresses

And also specific User Agent types, for example SIP scans from sipvicious which works if the User Agent header string is unchanged.

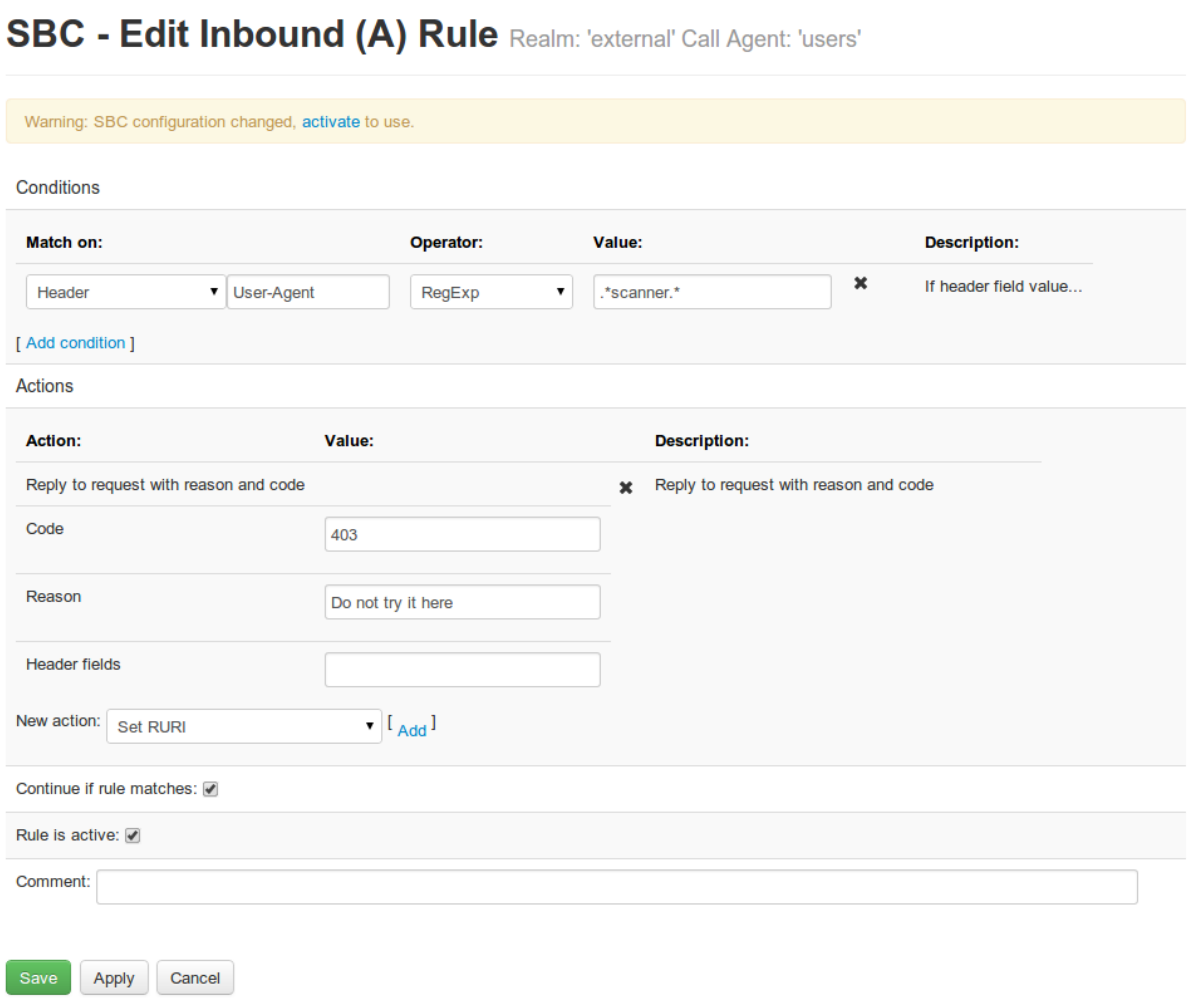


Fig. 19: Rejecting calls from certain user agents

4.3.11 Handling P-Asserted-Identity

The *P-Asserted-Identity* header is usually used within a network to signal the caller, if the identity is asserted, e.g. if it is signalled from a trusted source.

The *P-Asserted-Identity* header should usually only be trusted if it was set by some element in the internal network, e.g. by the home proxy after authentication. Hence, for requests coming from an external network it is recommended to remove the *P-Asserted-Identity* header*.

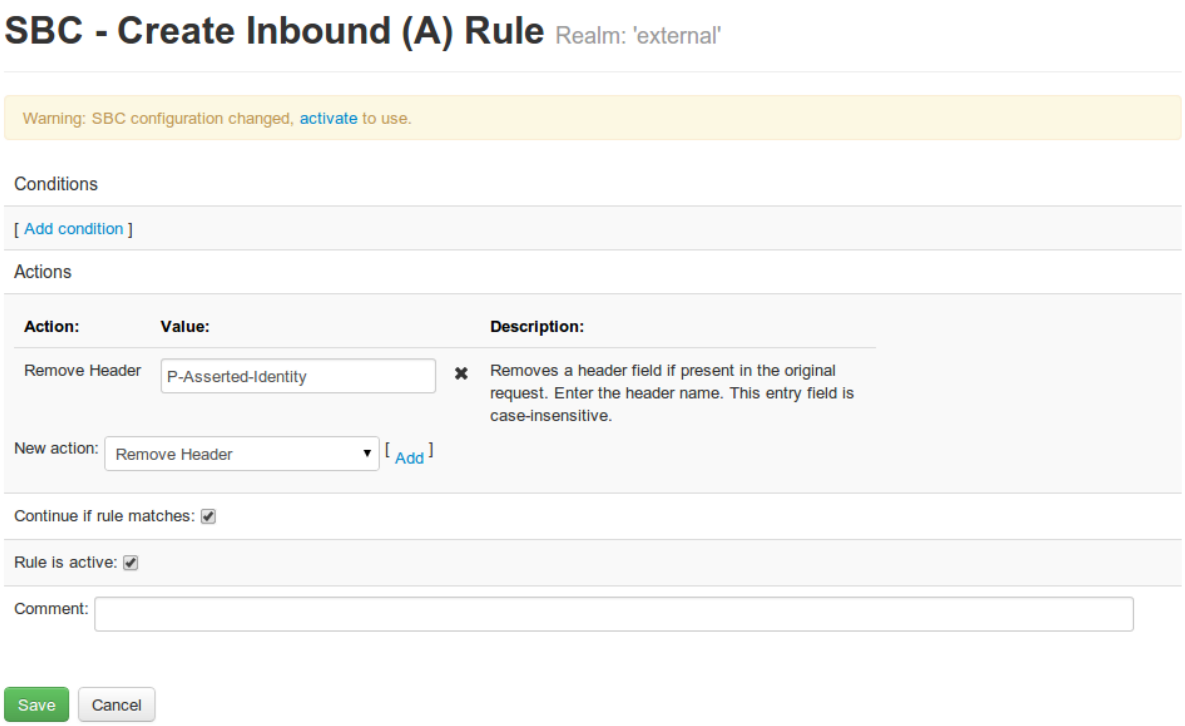


Fig. 20: Remove *P-Asserted-Identity* header from untrusted requests

4.3.12 Where to go from here

This section described a typical initial configuration for a simple use case and a simple network topology.

Going further from here, various use cases that are solved with the ABC SBC are explained in various sections of this document:

- Interworking with various types of PBXs requires often very specific SIP mediation actions which can be implemented using special rule sets, see *Defining Rules* and *SIP Mediation*.
- Quality of calls with the Enterprise trunking use case can be improved by using intelligent RTP relay handling, see *Media Handling* for more details.
- Mobile clients may benefit from specific codec handling and transcoding. See *Media Handling* for more details.
- For more security mechanisms, refer to chapter *Securing SIP Networks using ABC SBC and ABC Monitor (optional)*.
- Least cost routing can be implemented using Provisioned Tables. See *Provisioned Tables* for more details.
- For billing, the SBC can generate call data records (CDR). See *Call Data Records (CDRs)* for more details on how to use the CDRs and customize them.
- Both usage and the SBC host itself can be monitored through SNMP, see *Using SNMP for Measurements and Monitoring*.
- System administration tasks like backup, maintenance and upgrades are explained in chapter *ABC SBC System administration*.

Chapter 5

Installing the ABC SBC

5.1 Types of Installations: Hardware-based, Container and Cloud-based

The FRAFOS ABC SBC product is mainly distributed as a software based solution. The standard ABC SBC installation is done using rpm packages from software repository on top of clean Centos 7 operating system.

Another option is to install ABC SBC as a container, which can be run on operating system of customer choice, if the OS supports running of systemd type containers. Before using the container version, consider carefully the container functional and performance limitations.

FRAFOS also offers an Amazon cloud based solution where the ABC SBC and ABC Monitor is running as an instance in AWS (EC2). This is by far the fastest installation, the software can be started by several clicks. See Sections *Amazon Elastic Cloud Configuration Cookbook* and *ABC Monitor Installation Off AWS (optional)*.

FRAFOS can also provide a hardware based solution with preinstalled ABC SBC software on a reference hardware, see *Hardware Requirements* for more details.

5.2 Hardware Requirements

FRAFOS ABC SBC is based on CentOS 7 64bit operating system (x86_64 architecture).

Capacity and performance of the system depends mainly on the number and type of processors (CPU), available operating memory (RAM) and the number and performance of network cards (NIC).

There are no specific constraints for vendors of hardware and components, but we do have some suggestions and recommendations for the used hardware and its settings. Generally amount of memory and CPU power increases system resilience against load peaks, Ethernet cards with high packet rate facilitate high media anchoring throughput and fast solid-state drives facilitate WAV and PCAP recording.

Minimum hardware:

- CPU: 1x processor - 64bit architecture
- RAM: 4 GB
- NIC: 1x 1Gb network card
- HDD: 10 GB

Recommended / reference hardware: Fujitsu Primergy RX1330 (M3 or M4)

- CPU: Intel® Xeon® processor E3-1200 family, 4GHz
- RAM: 64 GB (DDR4 2666 MHz, dual channel)
- NIC: 2 x Intel 1Gb adapter

- SSD: 2 x 256 GB

For more details about system capacity and dimensioning, see Sec. *SBC Dimensioning and Performance Tuning*.

5.3 OS Tuning

This section only describes OS tuning options that – if chosen to be applied – must be performed` during the system installation. Various other parameters can be optimized later, see Section *Low-level Parameters*.

Correct hostname, and fully qualified domain name optionally, have to be set, as ABC SBC functionality depends on it. The current short and long (fqdn) hostname can be checked using:

```
% hostname
% hostname -f
```

The hostname and long hostanme have to be unique per ABC SBC node. If just short hostname is used, the command for long one should return also the short hostname.

If the hostname has to be changed, the following Centos command can be used:

```
% hostnamectl set-hostname <new_hostname>
```

5.3.1 Disk Partitioning

Note: this section applies to standard ABC SBC installation, not to container based install.

Depending on the configuration, running ABC SBC may be fairly disk-space-consuming. The ABC SBC collects all kind of diagnostics data (events, PCAP files), records and archives Call Detail Records and audio WAV files), maintains internal logs. To make sure that a configuration which produces extensive data will not interfere with vital system functions we recommend creating the following partitioning schema:

- / . . . for the OS and application. Recommended size at least 10GB.
- /var . . . Recommended size at least 30GB.
- /var/log . . . includes internal logs and CDR files. Recommended size at least 20GB.
- /data . . . includes traffic data such as recorded WAV-files, PCAP files and CDRs. Recommended size at least 50GB.
- no swap partition . . . For best operation the system should be equipped with enough RAM memory and no swap area used.

The recommended filesystem to be used is ext4.

5.4 Deployment Modes

According to the system dimensioning and high-availability requirements, ABC SBC can be deployed in different modes:

5.4.1 Single Node Mode

5.4.2 High Available (HA) Pair Mode

IMPORTANT note: up to ABC SBC release 4.1, it was using HA solution based on Pacemaker. The ABC SBC 4.2 was a transitional release that removed Pacemaker based HA solution. The new HA solution based on keepalived was introduced in ABC SBC 4.3 release.

HA pair ABC SBC installation is formed by two physically identical servers running in an active/hot-standby configuration. Only the active (HA master) server processes signalling and media traffic. In case of any failure, the internal management system performs a failover where the originally standby (HA backup) machine becomes active. Switching the active and standby operation modes is also useful during the upgrades of the system.

Both the HA master and backup servers share virtual IP addresses (VIPs) and communicate with each other over the "Internal Management Interface" - IMI. The HA backup server can check the availability of the HA master server. Once the HA backup server determines that the HA master server is no longer available then the backup server will assume the role of HA master and take over the VIPs used for receiving and sending the media and signalling messages.

Further, the HA master server replicates state information about running sessions to the HA backup machine. Thereby, after a failover the backup server will be able to continue processing already established calls and the failure of the server will not result in dropping of already established calls. Note however that calls are replicated after they are established and calls unanswered yet may be dropped. Also only signaling over connection-less transport protocols is certain to reach the Call Agents as transport protocol context gets lost during failover.

5.4.3 Cluster based solution

For very high traffic and performance requirements, ABC SBC instances (in a single or HA pair mode) can create a cluster. A SIP load balancer is put in front of these cluster nodes so as to distribute the SIP traffic to a particular ABC SBC instances.

5.5 Installation Procedure

The ABC SBC can be installed two ways:

- either standard installation from rpm package repository, performed on top of clean Centos 7 operating system
- or container based installation, using systemd type of containers, on top of operating system of customer choice

For the standard ABC SBC installation, refer to the following *Installation Procedure - Standard ABC SBC install* section.

For the container based installation, refer to *Installation Procedure - Container ABC SBC install* section.

5.6 Installation Procedure - Standard ABC SBC install

In this section we describe the installation process of the ABC SBC software from the software repository of FRAFOS on a hardware platform with a pre-installed CentOS 7 (64bit) operating system.

Before installing ABC SBC, install the CentOS 7 64bit (x86_64 architecture) operating system. Use only the latest CentOS 7 release. and install only the basic set of packages.

FRAFOS ABC SBC is installed as a set of rpm packages for CentOS from the FRAFOS repository. In order to connect to the remote repository, the server has to have a working Internet connectivity.

Note that if the ABC SBC is to be used in HA pair mode then the procedure described here will have to be conducted on both servers.

In case of problems please contact the FRAFOS support at support@frafos.com

5.6.1 Repository Access and Configuration

Before accessing FRAFOS software repository, customers must obtain valid certified SSL credentials from FRAFOS customer support. When working with security credentials, please use company name (in example bellow "COMPANY" – replace with your company's name) as file naming convention.

Client Certificate

Obtaining valid repository credentials is a three-step process: 1) the administrator must generate SSL credentials, 2) obtain certificate from FRAFOS customer support and 3) install the certificates.

STEP 1) To create the SSL credentials run the following command:

```
% # generate SSH credentials
% openssl req -new -nodes -days 365 -out COMPANY_req.pem -keyout COMPANY_key.pem
```

The command will prompt for the following information (make sure all parts are filled except optional one):

- Country Name
- State (optional)
- Locality Name (city)
- Organization Name (company)
- Organization Unit Name (optional)
- Common Name (company name)
- Email Address (company contact)
- Challenge password (used in case of certificate revocation)

When all data is entered, the command will create two files: certificate request COMPANY_req.pem and key COMPANY_KEY.pem.

Store the key in the expected location using the following command:

```
% mkdir -p /etc/frafos/certs
% cp COMPANY_key.pem /etc/frafos/certs
```

STEP 2) Obtain certificate from FRAFOS customer support

Submit the certificate request created in previous step (COMPANY_req.PEM) by email to FRAFOS customer support at support@frafos.com. In response to your request, you will be emailed a "tarball" named "frafos_abcsbc_repo.tgz" that includes the requested certificate.

STEP 3) Install the certificates

Unpack the tarball on the ABC SBC host:

```
% # extract certificate archive
% tar zfx frafos_abcsbc_repo.tgz
% cd frafos_abcsbc_repo
% # start script that will place the credentials in /etc/frafos/certs
% # and set rpm repositories
% ./frafos_abcsbc_repo.sh COMPANY
```

Important: The administrator has to copy the COMPANY_key.pem file to this "/etc/frafos/certs/" directory!

The certificate installation process is now complete. FRAFOS ABC SBC repository configuration is automatically set in "/etc/yum.repos.d/Frafos.repo" file.

FRAFOS Repository Initialization

Once the FRAFOS repositories are configured, the administrator has to update the CentOS package database on the system

```
% yum makecache
```

Install the "frafos-sbc-installer" package from Frafos repository using:

```
% yum install frafos-sbc-installer
```

FRAFOS provides three rpm repositories:

- ABC SBC - the main repository with the ABC SBC rpm packages.
- Mirror of CentOS 7 "base" rpm packages repository, which contains the whole set of CentOS 7 rpms (x84_64 and noarch architecture).
- Mirror of CentOS 7 "updates" rpm packages repository. The updates released by CentOS are available using this repository as soon as they are verified by FRAFOS to not brake any basic ABC SBC functionality.

To do the ABC SBC installation, you have to use the ABC SBC repository. We highly recommend to use also the two CentOS "base" and "updates" repositories provided by FRAFOS and to disable the original CentOS repositories, to ensure no new rpm update package is installed before it is verified by FRAFOS. To disable the original CentOS repositories, issue the following commands:

```
% yum-config-manager --disable "CentOS-7 - Base"
% yum-config-manager --disable "CentOS-7 - Debuginfo"
% yum-config-manager --disable "CentOS-7 - Extras"
% yum-config-manager --disable "CentOS-7 - Updates"
```

Make sure that no other rpm repositories than the "FRAFOS CentOS 7 - Base", "FRAFOS CentOS 7 - Updates" and "FRAFOS ABC SBC" are enabled. You can verify which rpm repositories are defined and which are enabled by command:

```
% yum repolist all
```

Make sure that the yum package manager is successfully updated with FRAFOS repository content ("frafos-sbc-...")

Important: If the package database is not successfully updated (e.g. "could not get frafos-sbc-... info" error message is shown), then contact the FRAFOS support.

5.6.2 ABC SBC Package Installation

Important: Please check Sec. *Hardware Specific Configurations* before continuing the installation.

Call the command to install ABC SBC:

```
% sbc-install
```

The installation process installs all required packages from CentOS or FRAFOS repositories. The process may take several minutes.

For specific cases, it is possible to use the following options when calling the sbc-install command:

- `--auto` Use automatic non-interactive mode, do not prompt for anything.
- `--force` Force the installation even if pre-installation checks fail.
- `--master` By default both the signaling "slave" Sbc part and configuration master part (providing gui) are installed. Use this option to install only the configuration master part.

- `--slave` Use this option to install only the signaling Sbc "slave" part, which will be managed from standalone configuration master.
- `--notryit` Do not install demo "tryit" webrtc web application, which is by default installed together with slave part.
- `--nodiag` Do not install diagnostic tools. The tools installed by default include: ngrep, tcpdump, gdb, psmisc, lsof, wireshark, bind-utils, sharutils, man, htop, traceroute, atop.
- `--noss` Do not install session store and restify Sbc parts. Those are by default installed together with slave part, but not running by default.
- `--nofw` Do not install Sbc firewall, ipsets and fail2ban related parts. Can be used when deploying Sbc e.g. inside container which has no support for iptables.
- `--noha` Do not install Sbc High Availability rpms.
- `--nodnsmasq` Do not install dnsmasq dns caching resolver, which is installed by default.
- `--withaws` Install also scripts and other stuff for installation on Amazon AWS.

For installation output details, administrator can check installation log files

```
% less /var/log/yum.log
% less /var/log/frafos/sbc-install.log
```

Important: In case of any error or problem during installation, contact FRAFOS support.

For any future updates, either of CentOS or of the ABC SBC, use only the "sbc-upgrade" command.

5.7 Installation Procedure - Container ABC SBC install

In this section we describe the installation process of the ABC SBC container version.

The ABC SBC container image provided is of systemd-nspawn container type. It can be unpacked into separate directory and started from there on operating system of customer choice, which has to be able to run the systemd type of containers, like Centos or Debian Linux (64bit "x86_64" architecture).

The exact way of how to start and manage the container depends on the operating system choice and the tools provided by it, like the "machinectl" command. This section lists just general examples and recommendations, based on Centos 7 OS.

The network mode to use is usually "host network", where the network interfaces are configured on the host server and are shared with the container, and the container itself cannot change any network system settings. Also the proper firewall rules have to be set on the host server, or firewall disabled on it (if firewall is either not needed, or there is separate firewall in the customer network).

It is highly recommended to use different hosts for SBC container and CCM container. An ABC Monitor container can be deployed on the same host as CCM container. The SBC and CCM, or SBC and Monitor containers cannot share the same host, unless network model is used which provides some form of networking / loopback isolation, to prevent port conflicts.

For the ABC SBC container to run properly, it is necessary to disable SELinux on the container host machine, if SELinux is enabled. This can be usually done by editing "/etc/selinux/config" file and setting "SELINUX=disabled" (takes effect on boot), plus running the following command to set it on the running system:

```
% setenforce 0
```

We assume example container name "testsbc" and image file "frafos-abc-sbc-4-6-1-481.tgz" used in the following steps. Note that the container name corresponds to the directory name and also is used by default as hostname for the started container. Note: as hostnames should not contain underscore character, better avoid using it in the name.

All the commands should be executed under "root" (or equivalent) user.

5.7.1 Unpack the container image

The ABC SBC container is provided in form of gzip tar file. After getting it from Frafos repository or file server, copy it to the target hosting server.

Create a directory for the container, on a partition with enough space. The recommended path is "/var/lib/machines/<name>":

```
% mkdir -p /var/lib/machines/testsbc
```

Unpack the container image into that directory. Make sure the correct tar options are used to keep all permissions, ownership and attributes:

```
% tar --xattrs -p --numeric-owner -C /var/lib/machines/testsbc \
  -xzf frafos-abc-sbc-4-6-1-481.tgz
```

5.7.2 Prepare dir for persistent data

This step is optional, but recommended. A separate directory, different from the base one containing the unpacked container, from the host server can be "mounted" to the container on startup and used for data that is expected to be persistent in case of container replacement (e.g. when replacing with newer version). This directory is seen as "/data" path inside of the container, and is used for some basic configuration files, traffic pcaps, recordings, backups etc. If separate directory from the host server is not used, the "/data" path is created just under the basic directory holding the container, and is lost if whole container is replaced.

Create directory for ABC SBC data under some host sever partition with enough disk space, like:

```
% mkdir -p /var/data/testsbc
```

Note: if more containers are expected to be run on the same host server, make sure each of them uses own separate directory for "/data". Do not use one common path for more containers.

5.7.3 Create systemd service file

The container can be started directly from command line using "systemd-nspawn -bD <dir>" command, but usually it is desirable to use a separate systemd service for it, which allows automatic start and better management of the running container.

Create a new systemd service file for the container, by creating file like "/etc/systemd/system/testsbc.service" with content like:

```
[Unit]
Description=Test Sbc Container

[Service]
ExecStart=/usr/bin/systemd-nspawn --machine=testsbc \
--directory=/var/lib/machines/testsbc/ -b \
--bind /var/data/testsbc:/data
Restart=always

[Install]
WantedBy=multi-user.target
```

Notes:

- If the persistent directory for "/data" was not created in the previous step, remove the "–bind /data/testsbc:/data" option.
- Adapt the "/var/data/testsbc" path to reflect the actual host directory used for persistent data.

- The "Restart=always" option makes the container to be restarted automatically in case it stops for whatever reason. It can be changed to "Restart=no" if needed.

Call command to update systemd services:

```
% systemctl daemon-reload
```

5.7.4 Start and manage the container

Use the following command to start the container:

```
% systemctl start testsbc
```

If it should be started automatically on host server boot, use:

```
% systemctl enable testsbc
```

To check status, the following command can be used:

```
% systemctl status testsbc
```

To list all running containers:

```
% machinectl list
```

Please refer to the machinectl man page for more information on how to interact with systemd-nspawn containers.

5.7.5 Connect to the container console

To connect to the container console, use the following command:

```
% machinectl login testsbc
```

It is also possible to connect to the container using SSH, if it is configured to be running there (depends on ABC SBC interfaces configuration and assigned "Apps"). On fresh ABC SBC container, the SSH is listening on port 24 by default. Same port is used for CCM only container.

The root user password is set to "frafos" on fresh container and it is required to change it during first login.

Next step is to perform the initial configuration, which is similar for both standard and container deployment types.

5.8 Initial Configuration

5.8.1 SBC Interfaces Overview

The ABC SBC uses five types of logical interfaces for management, signalling and media processing:

- XMI - External Management Interface - for GUI, SNMP, SSH, XMLRPC and other services that should be accessible from outside the setup
- IMI - Internal Management Interface - used for inter-node communication (in HA pair or cluster mode)
- SI - Signalling Interface - for SIP signalling (multiple SI interfaces can be configured)
- MI - Media Interface - for media (RTP/RTCP) processing (multiple MI interfaces can be configured)
- WS - Websocket Signaling - for SIP signaling over Websockets.
- CI - Custom Interface - for different applications depend on admin

Important: Before the following initial configuration, it is important to have all physical interfaces used by the SBC's logical interfaces configured and working (IP addresses and IP routing). Also, the hostnames of the machines have to be set, as they are used in initialisation scripts and for distinguishing SBC nodes.

5.8.2 Web GUI Configuration (Choosing the Cluster Config Master)

ABC SBC provides configuration GUI web interface. If there is more than one SBC node (in HA pair or cluster mode), the main configuration is managed from one selected node that acts as a configuration master, and all the SBC nodes automatically pull new configuration from it. The centrally configured configuration elements include ABC rules, Interfaces, Global Config Provisioned Tables, Realms, Call Agents, SNMP configuration data and Firewall Rules.

After ABC SBC installation, choose one (and only one) node which will be the main configuration node, and perform the following step on it. Perform this step also when configuring just one-node SBC setup.

```
% sbc-set-master
```

It will ask for the following:

- Choose if configuration master should be enabled on this node, select "Yes" to confirm that.
- Username and password for configuration gui admin access. New gui user will be created and added to "SBCadmins" gui group. Minimum password length is 8 characters and it must not contain spaces.
- Username and password that will be used to authenticate the nodes to the configuration master when performing config pull. Note that this is diferent username / password than for the gui access. The password must not contain spaces.

All ABC SBC nodes need to know which node acts as the main configuration node, to be able to pull automatically new configuration from it. The slave nodes will have only limited access to the administrative functions in GUI.

Perform the following command on all ABC SBC nodes, including the node selected to be the configuration master.

```
% sbc-init-config
```

It will prompt for the following settings:

- Address of the main SBC configuration master node. On nodes that are not configuration master use either IP adddress of the IMI (internal) interface of the SBC node acting as main configuration master, or dns name which resolves to it's IMI IP address. On the node being configuration master, localhost address (127.0.0.1) will be used automatically. If the node is going to be intentionally set to not use config master, use 'none'.
- Select if certificate of the configuartion master should be verified by this Sbc node when pulling config from it. For usual simple installation that uses the default automatically generated self-signed certificate choose No, for secured installation choose Yes. The CA certificate file (in PEM format) to verify config master can be optionally added in following prompt.
- The administrative domain. For usual installations just press Enter to use "default" administrative domain. For installation where administrative domains are used, enter name of the administrative domain this Sbc node belongs to.
- Username and password that is used to authenticate the access to configuration master when performing the config pull. Use the same username and password as used when a node was set to act as configuration master (using "sbc-set-master").
- Client certificate. For usual installations just press Enter to not use client certificate. For secured installation, where client certificate is used when pulling configuration from config master node, enter full path to filename with the client certificate for the ABC SBC node being configured. The file has to be in PEM format and has to include both certificate and key in one file. The configuration master will verify the client certificate if the option "Mandate and verify client certificate" is enabled on TLS profile assigned to the IMI interface of config master.

- Node uuid. For usual installations just press Enter and the node uuid will be generated automatically, or existing one used. If specific node uuid is required, enter it.
- Password for the "sbcsecuser" user. It is highly recommended to disallow direct root user ssh login to the ABC SBC node(s), which can be done later in gui under Global Config / Misc tab, and to use other user which is allowed to use "sudo" command to get root access. For this purpose, there is a "sbcsecuser" user automatically created during ABC SBC installation. When performing the initial configuration, the password for this user can be set.

5.9 Setting Up Web Interface Access and User Accounts

ABC SBC web interface is available at the IP address of XMI (external management) interface and can be accessed using https URL on port 443 (by default, can be changed) like this: https://192.168.178.178/

Note: initially, before ABC SBC is configured, the configuration GUI web interface is accessible on all ip addresses configured on the system. Later, after SBC interfaces are configured in the GUI, it will be restricted only to XMI interface address. The default port is 443 but can be changed when adding XMI interface if needed.

For the main ABC SBC configuration access the XMI IP address of the SBC node which was selected to act as main configuration node previously.

Further information about managing administrative users can be found in the Section *User Management*.

5.9.1 Default User Accounts

Starting with "SBC" 4.6 release, there are no default GUI users. The username and password for user with admin rights for GUI access has to be entered when performing initial configuration of the configuration master node (done using the "sbc-set-master" command). Then the GUI users can be managed via GUI - new users can be added or assigned to groups, as described in the section *User Management*.

Group membership defines privileges of the respective users. The following groups come preconfigured:

- **SBCadmins** SBC administrators having access to all configuration
- **SBCrevisor** Read-only access to everything
- **SBCrpc** Access to XML-RPC interface. **Note:** using this group standalone is useless. You should use it together with other group specifying which xmlrpc resources the user have access to.
- **SBCusers** access to SBC related configuration (no rights to system configuration - networking, users, firewall etc.)

5.10 ABC SBC License

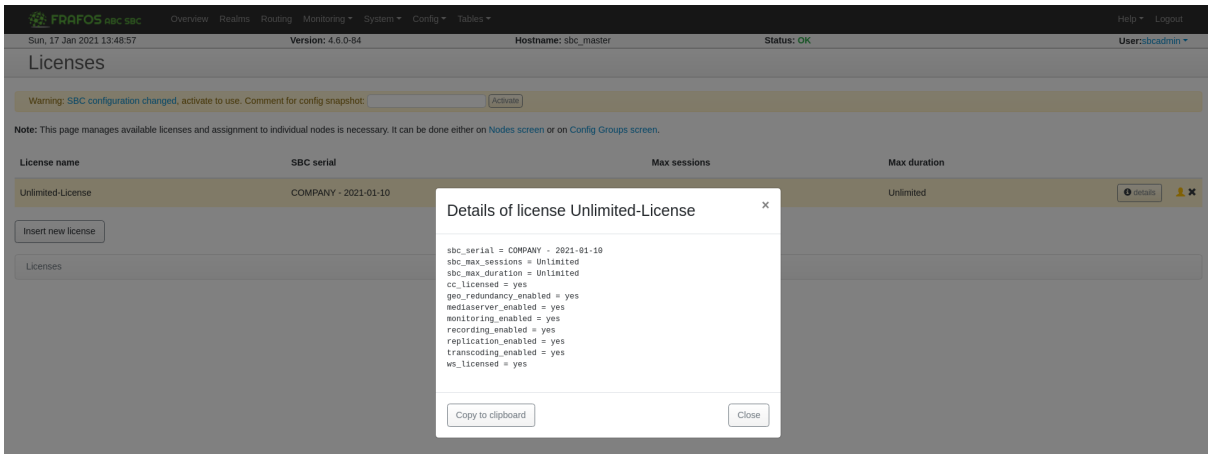
By default, the FRAFOS ABC SBC is installed in a demo version, which is limited to 90 seconds call duration, does not include support for replication, high availability and extension packages. Enabling these features requires a license file. FRAFOS issues license files according to the agreement between FRAFOS and the customer. The license file enables features as shown in the table bellow:

Licensing Package	*Feature*
transcoding	action: "Activate Transcoding"
recording	action: "Activate Audio Recording"
RTC	interface: "websocket"
media server	action: "refuse call with audio prompt"
high-availability	background active/standby replication
monitoring_enabled	gathering monitoring information for use in ABC Monitor

In the demo version without proper license set, the respective features are not executed. When the number of maximum calls is reached, the ABC SBC returns a SIP response "503 Server overload" and, if monitoring is enabled, issues a "limit" event with reason "licensed session limit reached". When the maximum duration is reached, the server terminates the call by sending BYE request to both parties, and if monitoring is enabled, issues a "call-end" event with originator field set to "internal-disconnect".

The license file has to be imported to the SBC using the 'System → License' link. Using the 'Insert new license' button, the administrator should give a name and selects the proper license file from the local disk by clicking 'Browse' button. After applying the changes, the license file is automatically uploaded to the server and loaded.

On Amazon Web Services, the paid-AMI instances download their license files when they start and no additional license configuration is required.



Important: For HA or a cluster deployment, the license file has to be imported on all nodes.

5.11 Interface Configuration

- *Physical and System Interfaces*
- *SBC Interfaces*
- *Retro Compatibility*

5.11.1 Physical and System Interfaces

System (network) interfaces can be configured either by manually editing the CentOS network configuration files located in */etc/sysconfig/network-scripts* or using standard Centos tools *nmtui* or *nmcli* if NetworkManager is being used.

Several types, like simple system network interfaces (e.g. eth1), VLAN tagged interfaces (e.g. eth1.100) or bonded interfaces (e.g. bond0) can be configured and used in the ABC SBC configuration.

IP Routing

If necessary, IP routing rules can be either manually changed in the configuration files located in the same directory as the interfaces configuration (/etc/sysconfig/network-scripts) or using standard Centos tools.

SBC nodes

If ABC SBC is installed in HA (active-standby) or cluster mode, the main configuration node should know about all the SBC nodes. This is required specifically in case the SBC interfaces settings differ between the nodes - e.g. when the nodes differ in IP address or system interface name used for one interface of the same logical type.

By default, each node has unique node uuid created locally at the node during initial configuration, and on configuration master side the node records are added automatically when the nodes pull configuration for the first time. The automatic adding of node records can be disabled under "Config → Global Config → Misc → Automatically add new nodes".

In case it is needed to add node records manually, either because automatic adding of node records is disabled or the records are needed to complete configuration even before the nodes try to pull configuration for first time, it can be done on the "System → Nodes" GUI screen of the main configuration master node. For each SBC node, you have to enter it's node name and node uuid. The node name field is just informational, and e.g. node hostname can be placed there. The node uuid is either generated on the nodes when performing the initial "sbc-init-config" configuration step, or if specific node uuid is required it can be entered manually when doing the initial node configuration. The node uuid is used to match the node to node specific settings.

Configuring Virtual IP (VIP) Address (OPTIONAL: in HA mode only)

When deployed in an HA active/standby mode two instances of the ABC SBC nodes will share one or more Virtual IP addresses. Virtual IP addresses are assigned to the currently active node.

The HA is configured using the "System → HA" screen.

For each pair of HA nodes a "HA group" can be created and the nodes assigned to it using "System → Nodes", or the HA group can be created also directly on the Nodes page while adding a new node. This HA group says which nodes will share the HA VIP IP addresses.

It is mandatory for the nodes in HA group to have IMI interface defined, and they must not use "IP autoconfig" option on the IMI interface, as the two nodes in HA group use the IMI interface IP address for HA "heartbeat" between the two nodes.

Under the HA group, you can add one or more VIP - Virtual IP addresses. For the VIP enter the IP address and optionally (recommended) also a netmask of the IP address. The netmask has to be in CIDR notation (like "24") or subnet mask (like 255.255.255.0). If netmask is empty, a mask "32" (meaning single host) will be used.

Optionally, also one or more HA routes can be added, which are bound to a particular VIP address. Such routing rules will be brought up and down together with the VIP address. For the HA route, the following data can be entered: Route destination - in form of subnet/netmask, like 192.168.0.0/24, this field is mandatory. Other fields are optional: Gateway - the routing gateway IP address, Source - the source address to prefer when sending, Table - table id if policy based routing is used.

Once the VIP address(es) are defined, it is possible to select to use VIP and choose particular VIP address when configuring ABC SBC interfaces using "System → Interfaces" screen. The VIP address can assigned to the SBC signaling, websocket or media type of interfaces.

5.11.2 SBC Interfaces

For signaling and management the ABC SBC uses six types of "logical" interfaces:

- **XMI - External Management Interface -** this interface is used for the GUI, SNMP, XMLRPC and SSH and other administrative services that should be accessible from outside of the CCM node. This interface must not be configured to be on a VIP. Default port numbers accessible on this interface are: 443 for ABC SBC GUI (can be set to other if needed), 1443 for XMLRPC provisioning access.
- **IMI - Internal Management Interface -** the IMI is used for inter-node communication (in HA pair or cluster mode) and for configuration transfer from configuration master to ABC SBC node(s). Only one IMI can be configured. Separate system interface using IP subnet not routed or accessible from outside should be used for IMI, unless there is a external firewall in front of ABC SBC. The port number accessible on IMI for the config pull from configuration master is 444. On single node setup it is not needed to create IMI interface and system loopback interface is used for the config pull. On more nodes setups it is mandatory to create the IMI interface.
- **SI - Signaling Interface -** SI is used for SIP signaling. Multiple SI can be configured.
- **MI - Media Interface -** MI is used for media (RTP, UDPTL, ..) processing and relay. Multiple MI can be configured.
- **WS - Websocket Signaling -** WS is used for SIP signaling over Websockets. This is useful only if the ABC SBC is configured to act as RTC gateway as described in Section *SIP-WebRTC Gateway*.
- **CI - Custom Interface -** CI is used for different applications which can be used for specific purposes like SSH, XML RPC, SNMP, Tryit webrtc client, TURN, HTTP proxy and HTTP redirect.

Signaling and media interfaces can be configured in different combinations. All SI/MI can share the same system interface, can be configured on a "per Call Agent" basis where each Realm has its signaling and media interface, or can share one assigned IP address with different ports per SBC interface.

It is also possible to create separate signaling and media interfaces on the same system interface for different purposes. For example, one for a PSTN gateway and one for receiving calls from residential users. In this case, a different signaling port and media port range shall be used. A typical ABC SBC configuration is to have one separate XMI, one separate IMI and one shared signaling and media IP address for each Realm.

When doing the initial ABC SBC configuration, add the XMI and IMI interfaces. The XMI interface has to be defined always on CCM node, to allow access to ABC SBC GUI. The IMI interface has to be defined always if HA or cluster mode is used (otherwise needed firewall rules would not be set).

Then add the interfaces for the SBC application: signaling (SI) and media (MI), optionally websockets signaling (WS).

If a specific application is needed, custom interface (CI) can be used with any port requested by admin.

When adding logical SBC interface, you first define it's name and options that are common to all SBC nodes using this interface, then you add records under the logical interface which map it to system interface for node(s) that will be using this logical interface. The list of records that map logical SBC interface to system interface on node(s) can be expanded or collapsed using the "+" or "-" icon before interface name. New mapping of logical SBC interface to system interface can be added using the "insert new system interface" button located at left hand side of the list.

In HA or cluster mode, if the interfaces differ between the nodes (use different IP address or system interface name), you have to create more separate logical to system interface mapping entries under the logical interface. Create a separate entry for each SBC node, set owner type to Node and select the node under Owner. Note: if records both for all nodes under a config group (owner type of config group selected) and for specific nodes are created, each node will use the record for all only if specific record for that particular node does not exist.

If the SBC interface settings do not differ between nodes, you can create just one logical to system interface mapping entry under each logical interface, set Owner type to config group and use the "default" config group.

If SBC interface is going to use VIP address (shared IP), the VIP address should be added before adding the interface.

SBC Interfaces are configured in the "System → Interfaces" screen.

The following parameters can be defined for logical SBC interface:

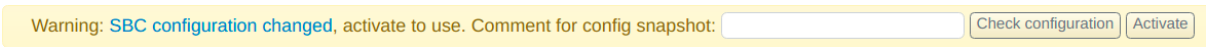
- Interface name: a unique identifier of the logical interface - [a-z, A-Z, 0-9].
- Interface type: Signaling, Media, WebSocket Signaling, External management, Internal management, Custom.
- Interface description: description (alias) for the interface that is used in the GUI configuration.
- TLS profile. By default, the TLS profile is set to None, meaning no TLS will be used on the interface. If TLS is to be used, select the TLS profile to use on the interface. The TLS profiles can be edited under System / TLS profiles page. There is profile named "default" which is automatically created at ABC SBC installation and uses self-signed certificate. Usually the default profile is used also when adding XMI interface for https GUI access to SBC.
- Applications (Apps): each logical interface can have one or more "Apps" enabled, which tune what service on which port will be listening on that interface, plus allow setting more specific option.

Please refer to - *Reference Application Interface Options* section for the Apps options details.

After creating entry for the logical SBC interface, add at least one logical to system interface mapping under it. The following parameters can be set for the mapping:

- Owner and Owner type: These list-boxes options set to which specific node the mapping of SBC logical interface applies. It can be assigned to a particular node as been pre-configured under **"System → Nodes"**, or all nodes belonging to a particular config group ("default" by default). Note: currently SBC supports only one common config group named "default", which can be used if the mapping applies to all nodes.
- System interface: system interface name (eth1, eth1.123 - VLAN tagged, bond1 - bonded interface)
- Type of IP address: use "manual" to manually specify the IP address, which is the default. If "autoconfig" is used, the first IP address from the corresponding system interface will be taken automatically. Use "VIP" to select one of VIP addresses, which can be configured in case of HA deployment mode under "System → HA" screen. Note: when configuring IMI interface of a node belonging to a HA group, the IP address type has to be set to manual.
- IP address: you can specify the IP address of the interface.
- Type of public IP address: use "manual" to manually enter the public IP address in the following field, which is the default. Use "Amazon autoconfig" to autodetected the public IP address. Current options of autodetection include Amazon EC2 cluster method.
- Public IP address: this parameter is optional. It allows to configure an IP address that will be used instead of the real or virtual IP address in SIP signaling (in case of the signaling interface) or media description (SDP; in case of a media interface). This is very useful to support near end NATs, e.g. Amazon EC2. Please refer to Sec. *Physical, System and SBC Interfaces* more details on the topic.

Important: When the SBC interfaces are configured, a warning message with a button to activate the new SBC configuration is shown in the GUI. No SBC interface changes are applied until the "activate" button is used. When the configuration changes are applied, all services using network configuration are restarted (e.g. SIP and RTP processes, SNMP daemon etc..). Note that this may cause service disruption.



If the XMI management interface is accessible via public Internet, we recommend to enable blocking IP address for repeated ssh or GUI login failed attempts, which can be done by enabling options "Blacklist IP addr for repeated ssh login failures" and "Blacklist IP addr for repeated gui login failures" under Config / Global Config / Firewall tab.

5.11.3 Retro Compatibility

Retro compatibility was introduced with the ABC SBC 4.5 because of increment of the json config version from *1.0* to *1.1*. The major change was the addition of interface applications, allowing a better transparency and tweaking of what is running where.

As the change brings some inconsistencies between the two config versions, a *retro-compatibility* module was developed. The purpose of that module is to, for every new config version to be deployed, check the current config (version 1.1) against the target node release, and when needed convert the json config to the older format.

While some basic changes are made under the hood (converting application interface options to older global config options - sshd port value for example), more complex changes are reported as an error.

The following table maps possible error situations with suggested solutions.

Common issues and fixes

Error message	Cause	Fix
json retro-compatibility 1.1 to 1.0: *[APP NAME] app not supported on older setup*	The application is enabled on an interface assigned to an node which does not support it (< ABC SBC 4.5)	See the list of unsupported applications in the section Applications. Disable the problematic application.
json retro-compatibility 1.1 to 1.0: *custom interface isn't retro compatible*	Custom interface is a new type of logical interface which was introduced in ABC SBC 4.5. This interface is not backward compatible.	Unlink the custom interface from the node or node's config group.
json retro-compatibility 1.1 to 1.0: *different value provided for the [PARAM] (app [APP_NAME]) Imported values: [VALUES]*	Different values assigned for options on a pre 4.5 node. Example: sshd port 22 on IMI, 23 on XMI	Use the same value for every application assigned to the faulty node. Example: set 23 for both IMI and XMI

Applications

Name from error	Description	Interface	GUI name
gominstrator	Perform administrator actions on a host (4.5). Please note this does not affect xmloredis service in pre 4.5 releases	Internal management interface (IMI)	*Management for host*
try-it	Enable the try-it feature (4.5)	External management interface (XMI) Custom	*Tryit webrtc client gui*
turn	Enable COTURN (4.5)	Custom interface	*TURN server for websocket*
http_proxy	Allow custom nginx proxy (4.5)	Custom interface	*HTTP proxy*
http_redirect	Allow custom nginx redirect (4.5)	Custom interface	*HTTP redirect*
webconf-api	Expose sems's webconf mgmt via RESTful json API (4.6)	Internal management interface (IMI)	*Local webconf API*

5.12 TLS profiles Configuration

The TLS profiles screen is used for manage TLS profiles used by SBC interfaces (see section *Physical and System Interfaces*). Each SBC interface can be configured to use different TLS profile.

The TLS profile takes effect only on those "Apps" enabled on the corresponding SBC interface, which support using of TLS.

5.12.1 TLS profile options

Table 1: TLS profile options

SSL certificate file	Select a file containing SSL certificate in PEM format
SSL private key file	Select a file containing key for SSL certificate in PEM format
Trusted CA certificates file	Select a file containing of trusted CAs in PEM format
Verify peer certificate	If checked, the SBC verifies TLS certificate of the peer against the trusted CA file.
Enable let's encrypt	If checked, no certificate, private key nor CA certificates are required. The ABC SBC will handle by himslef the completion of the ACME HTTP01 challenge. An email address is required to generate the cert. The one provided for monitoring is used (see monitoring parameters) Be aware that loopback and websockets interface have an undefined behavior on certificate acquisition / renewal. But the choice is leaved to the end user of using them together.
DNS	DNS domain associated to the node. The DNS is used to complete the ACME challenge on the let's encrypt side.

5.12.2 Certificate requirements

For the TLS certificates to be used with ABC SBC, the following requirements have to be met:

- The IP address or hostname for which the certificate is issued needs to to be listed in it's

SAN (Subject Alternative Name) field.

- The "serverAuth" should not be set in "extendedKeyUsage" field of the certificate for SBC node

(client) side.

5.12.3 Let's encrypt certbot

If the "Enable let's encrypt" option is used, an SSL certificate will be automatically requested and renewed using Let's encrypt certification authority.

A dedicated TLS profile should be created. The TLS profile **must** then be assign to a single node system interface (Interfaces screen > Logical interface > System interface). Once assigned, publish the new configuration.

The *gocertbot*, located on the ABC SBC cluster config, will parse the *tls_profile* SQL table and filter the LE's assigned target's nodes. A request will be made to LE, providing the node's DNS (provided on the profile). LE will then answer a challenge token to the gocertbot, whom will be followed to the nodes's *xmloredis* (:4242) process.

xmloredis will dump the token to */var/www/lets_encrypt/{TOKEN}*, and then start a custom nginx instance (configuration file: */etc/frafos/letsencrpyt/lets_encrypt.conf*, pid file: */run/nginx-certs.pid*). LE will try to authenticate the DNS by accessing *DNS/.well-known/acme-challenge/{TOKEN}*.

In case of success, a TLS certificate is delivered to the *gocertbot* process, whom will persist the new certificate in database, provoking a new "dirty" state.

To complete the process, re deploy the new SBC configuration, which will provoke the dump of the new certificate into the */data/sbc/tls/* directory of the target node. The *xmlrpc* command *reload_ssl* will then be run to hot reload sems's certificates.

Requirement

The "DNS" field has to be set to a DNS name that resolves to the public IP address of the ABC SBC target node where the corresponding TLS profile has to be used. The ACME challenge to verify ownership will be done automatically against it.

A valid email address need to be registered so it will be used to create an Let's encrypt account and receive email alerts (GlobalConfig > System Monitoring > email address). The from email address need also to be filled for email alerts.

Renewal

The certificate renewal will be attempted automatically 15 days before it's expiration. After certificate creation or renewal, a notification email will be sent to administrator email address set under Global config / System monitoring and new config has to be activated from ABC SBC GUI to propagate the new certificate to SBC nodes.

Limitation

At this time, ABC SBC's gocerbot only support **one** DNS for **one** IP AKA: **one** Let's Encrypt tls profile per system interface.

Loopback **and** websocket interfaces have undefined behaviour on Let's Encrypt certificate acquisition and renewal. The choice is leave up to the end user, but we don't recommend the assignment of those profiles to theses interfaces type.

Debug

Table 2: Debug options

process	good for	logs
xmloredis	up nginx instance, present LE token	*systemctl status sbc-xmloredis* You can also set *"dev": true* in */etc/frafos/sbc-xmloredis.conf*
gocertbot	request LE's for the challenge filter and persit the cert database	logs are outputed to the USER syslog facility, in /var/log/frafos/sbc.log

5.13 Hardware Specific Configurations

Depending on the hardware used for the ABC SBC deployment, there may be some fine-tuning needed to get maximum performance.

5.13.1 MariaDB Server Settings

ABC SBC cluster config master node (CCM) is using internally a local MariaDB database for configuration and monitoring data. It is recommended to optimize the performance of the MariaDB server. This is specifically useful in cases when frequent and many event records are expected to be stored.

The MariaDB server settings are normally applied automatically during the ABC SBC CCM node installation. In case the System administrator would like to tune the MariaDB server settings manually, options like the following should be added the into "[mysqld]" section of MariaDB server configuration file located in "/etc/my.cnf", before installing ABC SBC:

```
[mysqld]
query_cache_type=1
query_cache_size=10000000
innodb_flush_log_at_trx_commit = 2
innodb_file_per_table
innodb_flush_method=O_DIRECT
innodb_log_file_size=1G
innodb_buffer_pool_size=4G
```

The options can be tuned depending on the available host memory. If for example a low-memory (<16GB RAM) machine is used for trials without specific performance requirements, the last two lines should be commented or removed.

Note: In an active/standby deployment with two servers this step has to be done on each server.

Note about MariaDB users: the ABC SBC uses MariaDB "root" user for it's own databases creation and then "abcsbc-rw" and "abcsbc-ro" users for regular access to the ABC SBC databases. During the installation process, random passwords are generated and set for MariaDB users root, abcsbc-rw and abcsbc-ro for security reasons, and stored for later usage by ABC SBC application and upgrade scripts. In case system administrator changes manually the MariaDB root user password, he has to update also the stored password by editing file "/etc/frafos/sbc/db_pwd_power", as it may be used by ABC SBC upgrade scripts.

5.13.2 Network adapters

If the SBC is configured to work as an RTP media relay and a high number of concurrent calls is expected, a good choice of hardware is critical, specifically in terms of the used network adapter. RTP media traffic means high packet rate, with many small packets passing through. Some network adapters have suboptimal throughput under such conditions. Important things to consider when choosing a network adapter are:

- More receive and transmit packet queues are better. Each queue should be using separate interrupts.
- The adapter method of distributing packets to individual queues should include not only IP addresses into the "hash" calculation algorithm, but should include also IP packet port numbers. (Otherwise the traffic may end up in just one or two queues in case the SBC is communicating with only a small number of other devices on even just one IP address.)
- The adapter should be able to buffer packets received and issue interrupts only after some amount of the packets were received or some timeout. This can be usually configured using "coalesce" adapter options.

There is a global config section "Lowlevel" prepared to allow fine-tuning of settings related to network adapters. The settings are applied after the server is rebooted. The reference of the low-level configuration parameters can be found in Section *Low-level Parameters*.

System administrator can edit the settings depending on the particular hardware used. The settings are:

- Network interfaces on which a "receive packet steering" kernel feature should be enabled. Recommended setting is to enable it on network interfaces used as media interface.
- Ethernet adapter coalescing options and rx/tx ring parameters. These affect how many packets the adapter may buffer before issuing an interrupt. There is no recommended setting, as the values highly depend on the ethernet adapter used.
- Network interfaces on which the individual interrupts for receive and transmit queues should be statically bound to individual CPUs. If running on multi-CPU or multi-core platform, the recommended setting is to enable this option for all network interfaces used as media interface.
- Options to unload kernel modules for connection tracking or to disable connection tracking completely. Recommended setting is to stop connection tracking. However firewall rules used on the SBC have to be considered as those may need connection tracking active. Note: the default firewall rules that come with the SBC do not use connection tracking.
- Option to enable or disable automatic run of "mysqlcheck" command at end of server boot process. This command checks and repairs (if needed) MariaDB ABC SBC database tables. Default and recommended setting is to enable it.
- Options to specify system writeback setting. These allow to tune Linux kernel disk caching performance. On a system with big amount of RAM memory, it it recommended to limit maximum number of data system can cache before writing back to disk, to prevent high i/o peaks when writing large amounts of data back.

5.13.3 Configuration of SBC Number of Threads

The major processes of the ABC SBC are running under the name of **sems**. The number of SBC "sems" process threads affects the overall performance in terms of the maximum number of concurrent calls or maximum rate of calls per second supported by the ABC SBC. The optimal settings depend quite a lot on the number of CPU cores of the server used and also on the type of traffic being processed. As a general rule, for high number of concurrent calls including RTP media with relatively low calls per second rate lower numbers of threads performs better, while for high rates of calls per second with SIP only and no RTP media higher number of threads performs better.

The default value for the number of threads is 16. The recommended settings are:

- for SIP+RTP traffic use a number of threads equal to the number of CPU cores multiplied by 4
- for SIP only traffic (no media) use a number of threads equal to the number of CPU cores multiplied by 16

The number of threads can be configured under "Config → Global Config → SEMS". The setting takes effect using the "Apply button". It is not propagated to the peer machine if HA configuration is enabled until "Activate SBC configuration" link is used.

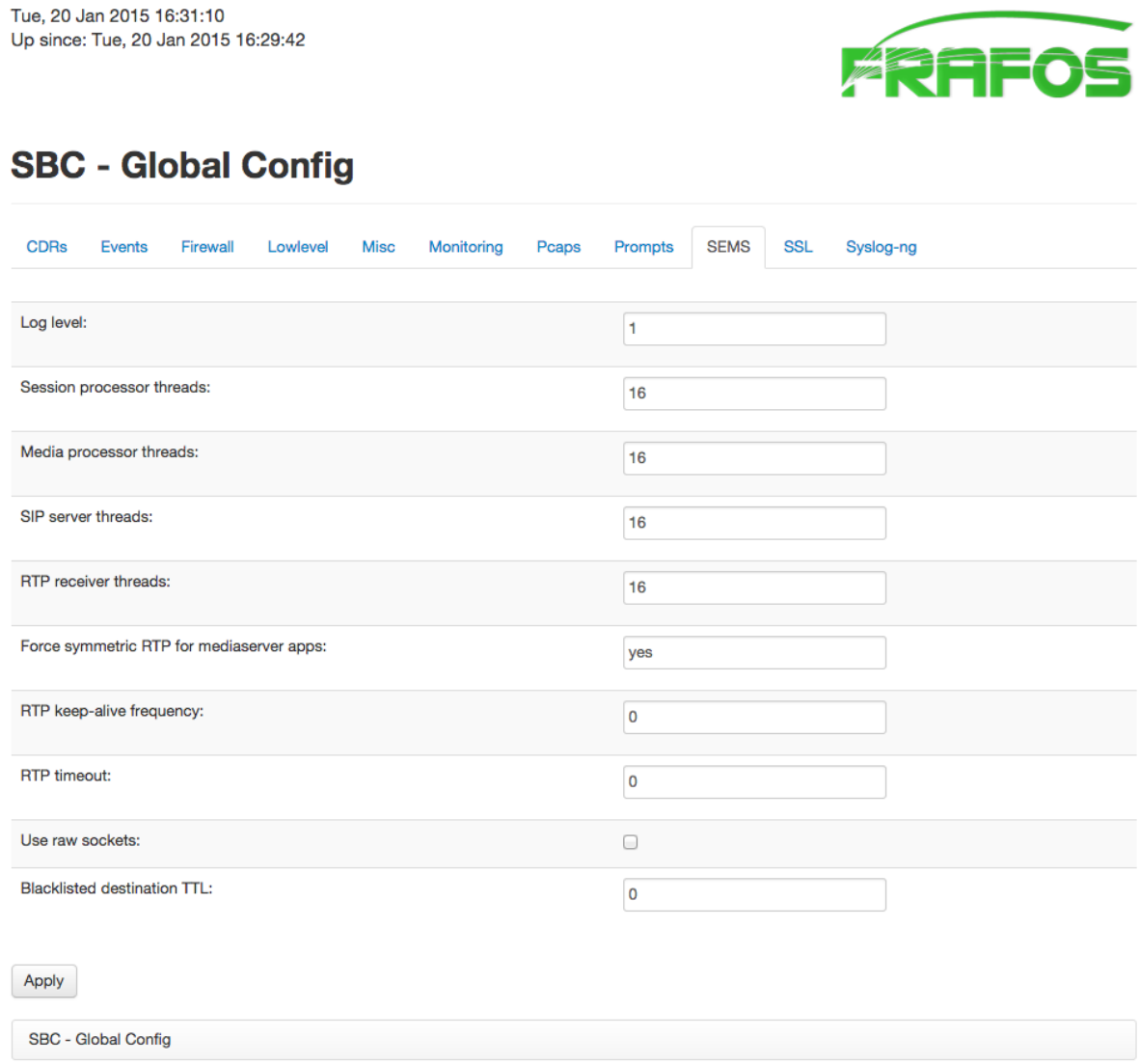


Fig. 1: Configuration of thread number

5.14 Last ABC SBC Installation Steps

Note: IMPORTANT: AFTER THE INSTALLATION PROCESS IS COMPLETE AND BEFORE CONFIGURATION AND TESTING BEGINS WE URGE YOU TO WHITELIST THE IP ADDRESS FROM WHICH THE ABC SBC WILL BE ADMINISTERED.

Failure to whitelist the administrator's IP address may – especially during the initial configuration and testing – easily block the administrative access to the machine. Various automated blacklisting techniques can block the whole IP address if they spot unexpected traffic from the IP address. See more details in Section *Automatic IP Address Blocking*.

To whitelist the IP address, visit the administrative GUI under **"Config → Firewall → Exceptions to automatic Blacklists → Add"** as shown in the Figure bellow:

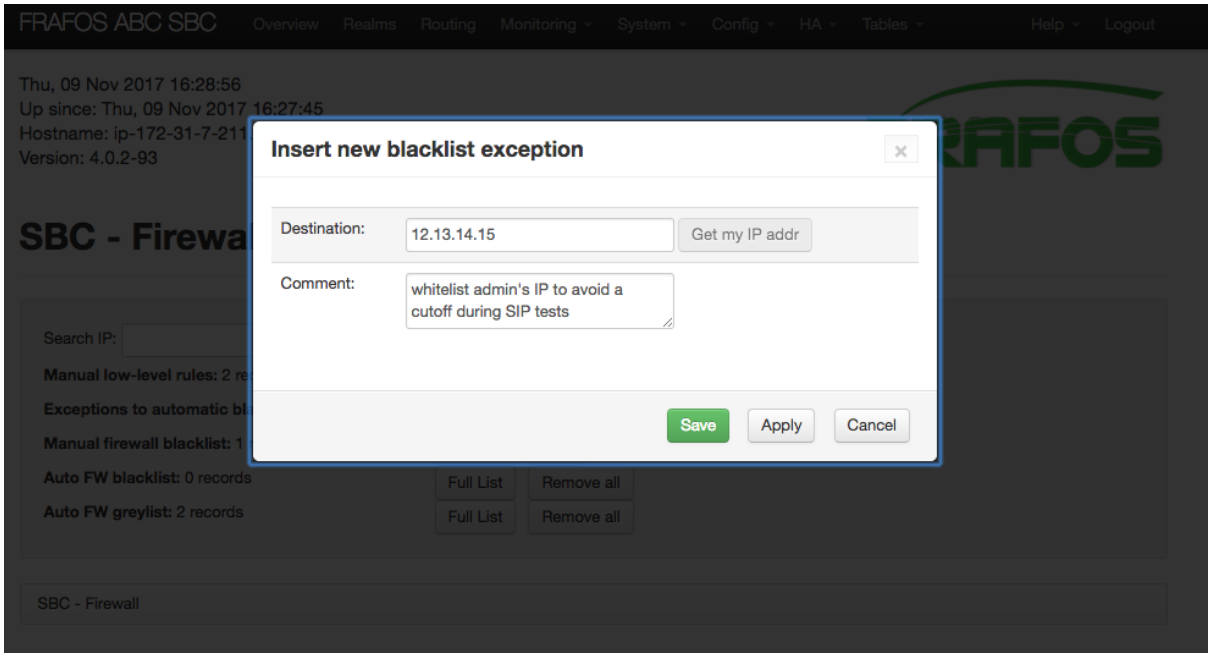


Fig. 2: Warning: Whitelist Administrator's IP Address

5.15 ABC Monitor Installation (optional)

In this section we describe the installation process of the ABC Monitor from the software repository of FRAFOS on a hardware platform with a pre-installed CentOS 7 (64bit) operating system.

The ABC Monitor is optional monitoring application, and if used, it has to be installed on a separate server. Running the server off the cloud is possible. If you would like to start the ABC Monitor off Amazon Cloud, skip this section and proceed to *ABC Monitor Installation Off AWS (optional)*.

FRAFOS ABC Monitor is installed as a set of rpm packages for Centos 7 from the FRAFOS public repository. In order to connect to the remote repository and for the installation process, the server has to have a working internet connectivity.

In case of problems please contact the FRAFOS support at support@frafos.com

5.15.1 ABC Monitor recommended server configuration

Minimum server configuration for ABC Monitor is:

- 1 x CPU with 4 cores
- 8GB memory
- 256GB disk
- 1 x 1Gb NIC interface

Recommended server configuration:

- 2 x CPU with 4 cores
- 32 GB memory
- 1TB SAS SATA SSD
- 2 x 1Gb NIC interface

Recommended disk partitioning:

- / 20 GB
- /var 20 GB
- /var/log 30 GB
- /data 1000 GB

Recommended file system to use: ext4.

We recommend to not use NetworkManager, which can be uninstalled using yum remove NetworkManager command, or can be disabled only on particular network interfaces using NM_CONTROLLED=no option in interface configuration files (/etc/sysconfig/network-scripts/ifcfg-*).

5.15.2 Repository Access and Configuration

Before accessing Frafos software repository, customers must obtain valid certified SSL credentials from Frafos customer support. Please refer to ABC SBC installation section *Installation Procedure* for the procedure to get and setup the client certificate to access the software repository.

FRAFOS Repository Initialisation

Once the FRAFOS repository is configured, the administrator has to update the package database on the system:

```
% yum makecache
```

Important: If the package database is not successfully updated, then contact the FRAFOS support.

Frafos ABC Monitor can be installed on clean CentOS 7 only, with only the following rpm repositories enabled from CentOS-Base: base, updates, extras. Make sure that no other rpm repositories than these basic ones and the Frafos repository are enabled in yum configuration files located in /etc/yum.repos.d/ directory.

5.15.3 ABC Monitor Package Installation

Install the "abc-monitor-installer" package from repository using:

```
% yum install abc-monitor-installer
```

Call the command to install ABC Monitor:

```
% abc-monitor-install
```

The installation process installs all required packages from standard CentOS or FRAFOS repositories. The process may take several minutes.

For installation output details, administrator can check installation log files:

```
% less /var/log/yum.log
% less /var/log/frafos/monitor-install.log
```

Important: In case of any error or problem during installation, contact FRAFOS support.

For future ABC Monitor updates and upgrades, use only the following command:

```
% abc-monitor-upgrade
```

5.16 ABC Monitor Container Installation (optional)

The ABC Monitor can be installed also as a container, running on top of operating system of customer choice, if the OS supports running of systemd type of containers. Please consider carefully the functional and performance limitations of the container version.

The installation steps are similar like when installing ABC SBC container version, please refer to *Installation Procedure - Container ABC SBC install* section for details. The important difference is that default SSH port of ABC Monitor container is set to 28.

Note: it is also recommended to use separate host server directory for the "/data" container path holding persistent data.

5.16.1 ABC Monitor Initial Configuration

After ABC Monitor installation is done, the password for user to access the ABC Monitor GUI has to be set. Use the following command to set the password for "sbcadmin" GUI user:

```
% abc-monitor-password
```

After the password is set, the ABC Monitor GUI can be accessed using web browser at https://<ABC Monitor IP addr>/. Use username "sbcadmin" for admin access.

In case the ABC Monitor is deployed as container, the default GUI https access port is 445, so has to be accessed at https://<ABC Monitor IP addr>:445/.

For initial ABC Monitor configuration, access the GUI on https (port 443 by default) and set some basic settings under the Settings / General page.

PLEASE SET at least memory options for elasticsearch and logstash processes, to correspond to total available system memory.

- Enable receiving of events from 4.1 SBC: The ABC Monitor release 4.3 and up is able to accept events coming from ABC SBC release 4.1 and up, but the communication channel for events from ABC SBC release 4.1 is different from 4.2 and later. Disabled by default. Enable only if receiving of events from 4.1 SBC is needed.
- Enable receiving events only via encrypted input: by default, ABC Monitor accepts events and replicated files (pcaps, recordings) coming both via normal unencrypted channel (which is default and has better performance) or encrypted using TLS. The way events are sent either non-encrypted or encrypted is configured on ABC SBC side. If this option is enabled, the ABC Monitor will accept only events coming via the TLS encrypted channel.
- Peer certificate verification level, if TLS used: if the events are being sent to ABC Monitor encrypted using TLS, this option sets the level of verification of the ABC SBC side certificate. Possible values are: 0 - ignore peer certificate, 1 - verify peer certificate if present, 2 - verify peer certificate, 3 - verify peer with locally installed certificate, 4 - ignore CA chain and only verify peer certificate. Default value is 0.
- Monitor GUI https port: sets the https port on which ABC Monitor GUI is accessible. Default value is 443.
- Number of days to keep old events before deleting them: this sets the time period, for how long ABC Monitor will keep the monitoring data collected from ABC SBC nodes. The data is stored in /data/elasticsearch directory. Default setting is 30 days. Amount of disk space used highly depends on the monitored nodes traffic. This setting is used also to delete old log files.
- Minimum partition free space percentage before deleting old events: if the free space on the partition where events are stored (that is partition holding /data/elasticsearch directory) drops below the value set, oldest events will be automatically deleted until the free space percentage is again above limit. Default value is 20. Set to 0 to disable the automatic deletion based on free space. Note: it is highly recommended to use separate partition for the /data/elasticsearch directory, otherwise old events may be deleted even if something else occupied the disk space. On SSD disks it is recommended to keep about 20% of free disk space for better performance and longer disk life.

- Time in minutes to keep old traffic pcap files before deleting them: this sets retention period for traffic pcap files that are synced from ABC SBC nodes to ABC Monitor, if it is enabled in ABC SBC global config. The setting must be greater or equal than the retention setting on SBC. (**"Global Config → Events → Number of days to keep old traffic log files"**), failure to dimension the period correctly may cause ABC SBC to keep uploading the files repeatedly. The data is stored in /data directory.
- Time in minutes to keep old recordings files before deleting them: similar setting like the previous one, but for recordings file, if replicating enabled in ABC SBC global config. The setting should be also equal to corresponding setting on ABC SBC. The data is stored in /data directory.
- Firewall: source IP addresses or subnets allowed to access Monitor GUI over https: limits access to ABC Monitor GUI using system iptables rules only from specified list of IP addresses, IP subnets or hostnames. You can specify more items separated by spaces. For IP subnets, use the CIDR notation like 192.168.0.0/24. It is highly recommended to limit access only from particular addresses or network subnets, to minimize security risks.
- Firewall: source IP addresses or subnets allowed to access Monitor over ssh: similar to the previous setting, but limiting access to ssh daemon.
- Firewall: source IP addresses or subnets allowed to push events to Monitor: limits access to pushing events from ABC SBC to ABC Monitor only from specified list of IP addresses or subnets. More items can be separated by spaces. It is highly recommended to limit access to ABC Monitor only from particular addresses, to minimize security risks.
- Monitor name: sets ABC Monitor name that will be displayed in GUI dashboards heading.
- E-mail TO for alerts: sets recipient e-mail address for sending alerts. Use empty value to disable sending of alerts.
- E-mail FROM for alerts: sets e-mail From address to be used when sending e-mail alerts.
- E-mail address for reports: sets recipient e-mail address for sending reports. (Note: the reports are not existing yet in ABC Monitor 4.3 and 4.4 releases, this option is prepared for later releases.)
- Logstash heap memory size: sets the amount of memory to use as heap for the Logstash process, which is the ABC Monitor part receiving and filtering events. It is recommended to use 20% of available system memory. Use "m" or "g" suffix to specify value in megabytes or gigabytes.
- Elasticsearch heap memory size: sets the amount of memory to use as heap for the Elasticsearch process, which is the ABC Monitor database part storing events. It is recommended to use 40% of available system memory. Use "m" or "g" suffix to specify value in megabytes or gigabytes.
- Logstash persistent disk queue size: sets the size of disk queue used as buffer for events. It is recommended to use at least 1GB of disk space. Use "mb" or "gb" suffix to specify value in megabytes or gigabytes.

The ABC Monitor GUI access supports two authentication methods: basic http authentication or LDAP authentication. Both are independent and may be enabled at the same time. In such case, the basic authentication is tried first.

The basic http authentication uses username "sbcadmin", and password can be set using "abc-monitor-password" command from command line.

On the ABC SBC side, to enable pushing of monitoring data to the ABC Monitor, you have to configure the ABC Monitor IP address in "ABC Monitor address" setting of Global Config, under Events tab.

By default, the pcap and traffic recordings files are copied from ABC SBC nodes to ABC Monitor using basic rsync protocol, which works without any extra configuration. If needed, rsync over TLS can be used if enabled in Global config on ABC SBC side.

The https access to ABC Monitor GUI uses automatically created self-signed certificate by default. If real trusted certificate is going to be used, it can be imported using the following steps on the ABC Monitor server:

- remove old certificate and key files, which are symbolic links pointing to the automatically created self-signed cert and key by default:

```
% rm /etc/ssl/certs/nginx.crt; rm /etc/ssl/private/nginx.key
```

- copy the new certificate file to /etc/ssl/certs/nginx.crt and key file to /etc/ssl/private/nginx.key
- restart nginx service:

```
% systemctl restart nginx
```

5.17 ABC Monitor LDAP Installation (optional)

If the LDAP authentication is going to be used, the following options can be set in ABC Monitor GUI Settings / General page. Alternatively, in case the ABC Monitor GUI is not accessible yet, the same options can be set from command line using "abc-monitor-ldapconfig" command.

- Enable LDAP authentication: if enabled, the authentication against LDAP server will be done.
- LDAP server address: the LDAP server address, use a format like "ldap://IP:PORT", "ldap://IP" or "ldap://my.domain".
- LDAP bind distinguished name: LDAP admin bind domain name.
- LDAP bind credential: admin password to authenticate against the LDAP server.
- Base DN of the LDAP server users: specifies the default base domain name. For "cn=admin,dc=example,dc=org" the base DN would be "dc=example,dc=org".
- LDAP extra groups: extra group to concatenate to the base dn to authenticate a user. We would set 'ou=People' to authenticate user with dn in the form "uid=john,ou=People,dc=example,dc=org"
- LDAP's group users need to belong to: a potential LDAP group, from whom a user wishing to log itself need to belong to. Please not that a full dn is expected, in the form "cn=GUI,ou=Groups,dc=example,dc=org".
- enable TLS LDAP: if enabled, the connection to LDAP server will use TLS

encryption.

- enable compatibility with Microsoft Active Directory

A testing docker instance may be found at https://github.com/frafos/docker-ldap. Use with the following options value : - enable ldap: true - ldap host: ldap://127.0.0.1:389 - bind dn: *cn=admin,dc=example,dc=org* - bind password: *admin* - search dn: *dc=example,dc=org* - extra groups: *ou=People* - user groups: *cn=GUI,ou=Groups,dc=example,dc=org* - ldap tls: false - ldap win: false

5.18 ABC Monitor Installation Off AWS (optional)

Starting the ABC Monitor off the AWS cloud is very fast, because ready-made virtual images already contain an installed ready-to-run system. An AWS-powered ABC Monitor can be used for trials, as a secondary Monitor when two are needed, or even as a primary system for both AWS and on-premises SBCs.

Before you start you will need the following:

- Amazon Web Services (AWS) account. Note that the accounts come with several service plans charged at different levels, and credit card number and a telephone must be ready to verify identity and payment. Go to http://aws.amazon.com to sign up.
- AWS Elastic Cluster SSH keypair. This is important to be able to administer the virtual machines remotely. If you haven't created or uploaded one, do so under "EC2→Keypairs". If you want to start the services in multiple regions, make sure that you have a keypair for every region before you start.
- SBCs with enabled monitoring license to provide the actual monitoring data.

To start a monitoring instance proceed as follows: - Start the instance.

- Visit https://monitor.frafos.com to start a proper AMI.
- Choose a properly dimensioned instance type with at least 8 GB of memory, such as M4.LARGE.

- Apply a proper security policy. Make sure you limit monitoring traffic to that coming from your SBCs and permit only port numbers 1873 and 16379. Administrator machines shall be allowed to access remote shell (port 22) and the actual monitoring GUI (port 443).

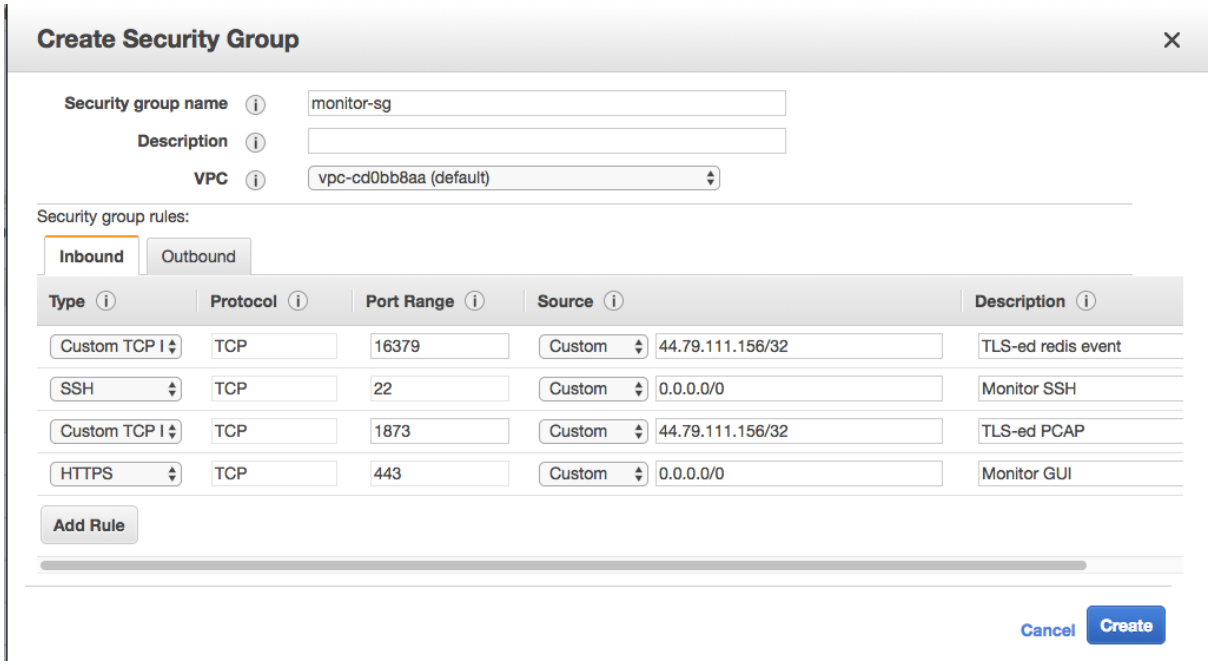


Fig. 3: An example Security Group for ABC Monitor Running on AWS

- Configure the ABC SBC to use the ABC Monitor instance as primary or secondary monitor.
  - open ABC SBC Administrative interface, the section "Config → Global Config → Events"
  - set "ABC Monitor Address" or "Secondary ABC Monitor Address" to the IP address of the ABC Monitor instance. If on the same net, use the private IP address, use the public IP address otherwise.
  - optional: turn on the checkbox "replicate traffic logs to ABC monitor" or "replicate traffic logs to secondary ABC Monitor".
  - optional: turn on the checkbox "replicate recordings to ABC monitor" or "replicate recordings to secondary ABC Monitor".
  - turn on "Use TLS secure connection to ABC Monitor" and set "Verify level for TLS connection to ABC Monitor:" to zero
  - apply the changes
  - activate the changes
- Access the ABC Monitor: visit https://IP, use "sbcadmin" as username, instance-id as password

5.19 G729 Codec Installation (optional)

G729 codec is by default **not installed** together with the rest of ABC SBC.

If it is needed, administrator has to explicitly install `sems-g729` package on all SBC nodes in cluster that should be using it.

Without installing this package, functionality like transcoding into G729 will not work. In such case configuration trying to use G729 for transcoding won't be accepted by the appropriate SBC (will be shown in Node's status in ABC SBC GUI).

It is highly recommended to update ABC SBC to latest version before installing the codec package:

```
% sbc-upgrade
```

After that being successfully done the codec can be installed:

```
% yum makecache
% yum install sems-g729
```

Please note that installing the G729 codec package may cause service disruption because new configuration file for SEMS process needs to be generated and SEMS process restarted to apply the change.

If the codec is not needed any more (it is not referenced by the configuration any more), the `sems-g729` package can be removed from the SBC:

```
% yum remove sems-g729
```

Please note that its removal may cause service disruption again: SEMS configuration file must be re-created and SEMS process restarted again.

Chapter 6

General ABC Configuration Guide

6.1 Physical, System and SBC Interfaces

In the ABC SBC we distinguish between physical, system and SBC interfaces, see the Figure *ABC SBC Interface defintion*:

- A physical interface is one of the network interfaces (cards) physically available on the system.
- System interfaces is an interface mapped on one or more of the physical interfaces. A system interface can be a "simple" physical interface (e.g. "eth2"), a VLAN (e.g. "eth3.1") or a bonded interface that is bound to two physical interfaces (e.g. "bond0" created by bonding "eth0" and "eth1" physical interface).

  For active/hot standby high-availability mode, it is highly recommended to use bonded interfaces with each physical interface connected to separate L2 switch to ensure reliable physical connections.
- SBC interfaces: These are logical interfaces used by the ABC SBC in order to distinguish between management, signalling and media traffic.

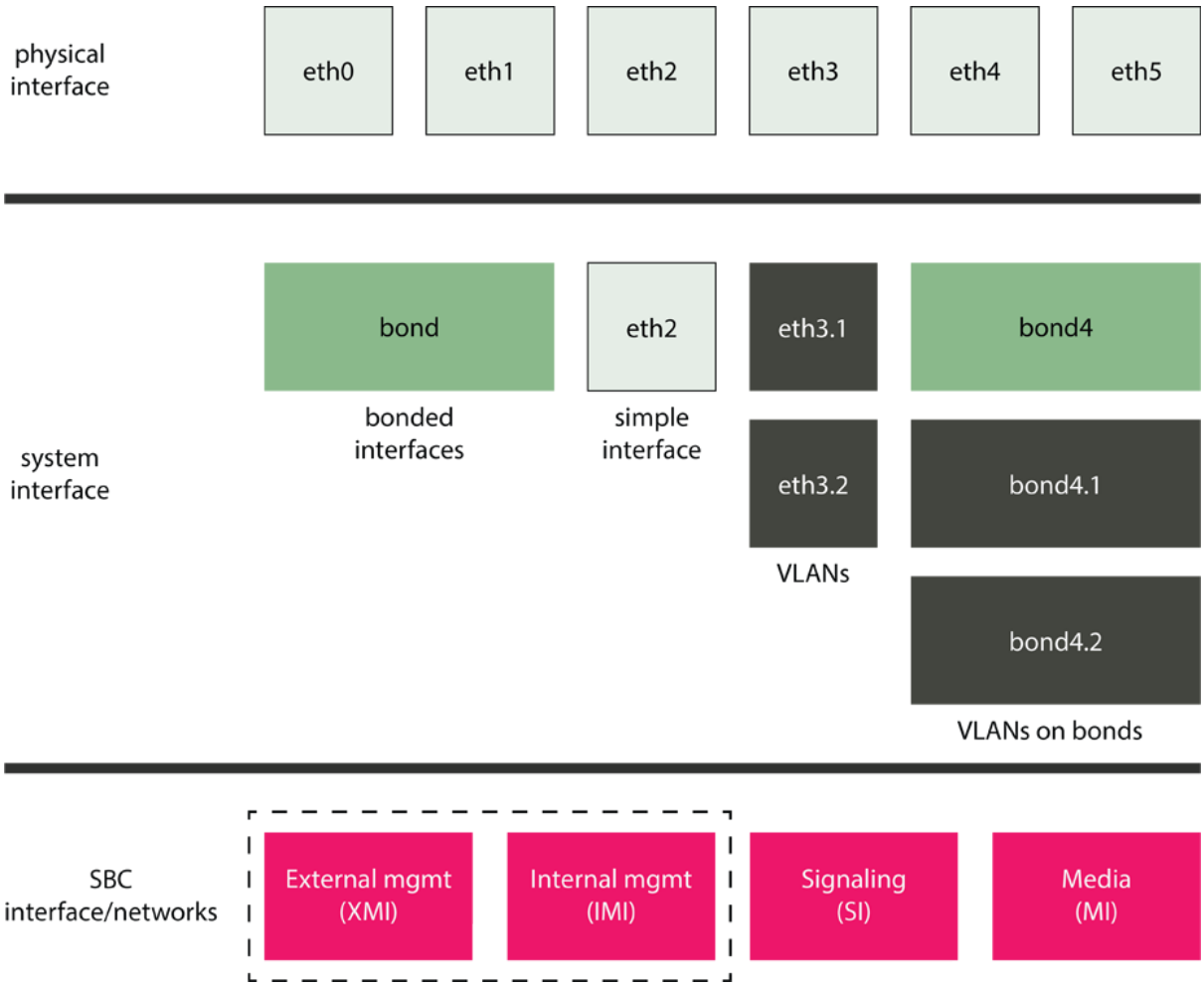


Fig. 1: ABC SBC Interface defintion

For the details on the configuring the interfaces see Section *Interface Configuration*.

6.2 Defining Rules

The ABC SBC's behavior is specified in form of rules, as explained in Section *A-B-C rules*. These rules consists of conditions and actions and are processed sequentially until a matching rule is found.

Each rule may have none, zero or multiple conditions. If no condition is specified, the rule always matches and all its actions are executed. If multiple conditions are associated with a rule, the rule matches only if all of the individual conditions match.

An example of such a rule is shown in the Figure *Example Rule*. It consists of two conditions that match if a call is intended for a telephone number beginning with 900 and the caller is not registered. If the condition applies, the call is rejected using the 403 SIP code.

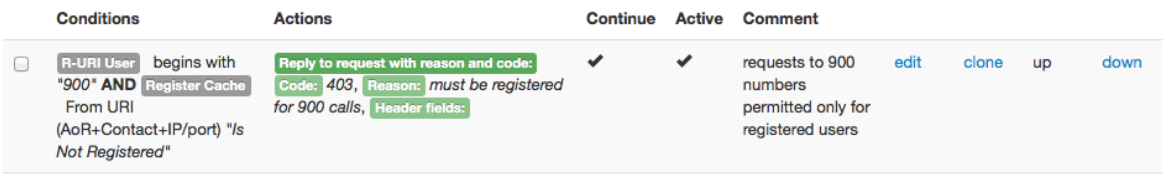


Fig. 2: Example Rule

Each individual rule condition consists of three parts: a condition type, an operator and a value. Subsequent subsections describe all these parts in details.

6.2.1 Condition Types

The type of a condition defines the left operand for the operation. The following table describes all available condition types and operators types that are applicable to the respective condition types. Operators "==", "!=", "RegExp", "does not match RegExp","begins with", "does not begin with", are supported unless specified otherwise.

Table 1: Condition Types

Condition type	Description
Source Call Agent	Check the source call agent. Only operators == and != are supported.
Source Realm	Check the source realm. Only operators == and != are supported.
Source IP	Check IP address the incoming request was sent from.
Source port	Check port number the incoming request was sent from.
Inbound interface	Check local interface the incoming request was received on. Value has to be chosen from a list of configured signalling interfaces. Only operators == and != are supported.
Source-IP CC (GeoIP)	Check country code against source IP's geographic region. Note: the geoip database must exist, which needs providing geoip license created using customer account at MaxMind, via global config option under Config / Global Config / Misc tab.
Destination Call Agent	Check the destination call agent. Only operators == and != are supported and only available in realm C rules.
R-URI	Check current request URI.
R-URI User	Check user part of request URI
R-URI User Parameter	Check parameter in username part of request URI. For example in the R-URI "sip:106;name=franta@domain.com", the parameter "name" can be checked for value "franta"
R-URI Domain	Check host part of request URI, can contain port number
R-URI URI Parameter	Check parameter of request URI.
From	Check From header field value.
From URI	Check value of From URI.
From User	Check user part of From URI.
From Domain	Check host part of From header URI, can contain port number
To	Check To header field value.
To URI	Check value of To URI.
To User	Check user part of To URI.
To Domain	Check host part of To header URI, can contain port number.
Supported header	Check content of Supported header (since version 4.5)
Require header	Check content of Require header (since version 4.5)
Header	Check value of given SIP header or test if a SIP header does not exist. This condition is a kind of "escape-code" for testing headers for which no other conditions exist. The following header-fields will not be processed using this condition: From, To, Call-ID, CSeq, RAck, RSeq, Route, Contact, Via, Max-Forwards, Record-Route, Content-Type, Content-Length
Codecs	Check presence/absence of codecs within SDP. Right operand specifies codec name. Only operators Contain , Contain RegExp and Do not contain are supported.
Media Types	Check presence/absence of media type within SDP. Right operand specifies media type name (e.g., audio,video) Only operators Contain, Contain RegExp and Do not contain are supported.

continues on next page

Table 1 – continued from previous page

Condition type	Description
SRTP Types	Check presence/absence of SRTP type within SDP. Right operand specifies media type name (e.g. dtls, sdes, none) Only operators Contain, and Do not contain are supported.
Call Variable	Check call variable value using selected operator. The call variable has to be already defined by Set Call Variable action. Any condition referring to an undefined value returns FALSE as result.
Call Variable Existence	Tests if a variable exists or is undefined. This is useful for example when table lookups are used to discriminate accurately between non-existing and empty values.
Generic Text Match	Compare two generic text expressions, supporting replacements.
Method	Check SIP request method. Value has to be chosen from a list of allowed methods. Only operators == and != are supported.
Register cache	Check content of register cache. Operands From URI (AoR + Contact + IP/port), From URI (AoR + IP/port), Contact URI (Contact + IP/port), To URI(AoR),R-URI (Alias) are supported.
NAT	Check first Via address whether the sender is or is not behind NAT. This compares the SIP message source IP with the first Via address and works only if the UA directly communicates with the SBC.
Read Call Variables	Trigger a predefined query in provisioned tables by a specified key value. The condition returns true if the lookup was successful, false otherwise.
Last Action Result	Returns true if the last action completed successfully, false otherwise (analogical to shell $? variable).
Blacklist	Checks if a call-agent is on a black-list (or not). A call-agent is blacklisted when it is not reachable to make sure that no futile attempts to send traffic to it are undertaken.
Date and Time	Checks whether a Datetime, Date or Time value is before or after now plus an optional offset. The value may be in the form '2016-05-25 13:10:41', '2016-05-25' or '13:10:41'. The offset can be years, days, hours, minutes or seconds, e.g. '2y', or '30d', or '12h', or '5m', or '1800s'.
Parallel Call Count	Tests if the number of parallel calls is below or above a threshold. The number refers to the specific place in rules execution flow from which the condition was evoked. It does not refer to a global number of calls.
Parallel Call Count (global key)	Tests if the number of parallel calls for a global key is below or equal/above a threshold.
Request source	Tests whether request being currently processed is generated by the SBC itself, alternatively as a result of unattended call transfer.

6.2.2 Condition Operators

Operators supported within general conditions:

Table 2: Condition Operators

Operator	Description
==	left operand equals given value
!=	left operand does not equal given value
RegExp	left operand matches given regular expression
does not match RegExp	left operand does not match given regular expression
begins with	left operand starts with given string
does not begin with	left operand does not start with given string
Contain	right operand is contained in
Contain RegExp	sample described by right operand is contained in
Do not contain	right operand is not contained in

It is important to know that if a mediation action (Section *SIP Mediation*) changes content of SIP message, the

condition will refer to the value **after** modification. E.g., if you apply the rule action "SetFrom(sip:new@from.com)", the "From URI" operator will return sip:new@from.com!

Some conditions types take special operators and/or values. Particularly the "Register Cache" condition tests if a registration can be found in SBC's cache. The condition uses a specific operator that determines which URIs are used for the test.

Supported operators for "Register Cache" are:

Table 3: "Register Cache" Operators

Operator	Description
From URI (AoR + Contact + IP/port)	the user with given From URI and Contact is registered from given IP:port
From URI (AoR + IP/port)	the user with given From URI is registered with any Contact from given IP:port
Contact URI (Contact + IP/port)	a user with given Contact is registered from given IP:port
To URI (AoR)	the user with given To URI is registered
R-URI (Alias)	the user with given request-URI is registered

The value for the "Register cache" condition allows to refine the test. It can be one of the following:

Table 4: "Register Cache" Conditions

Condition	Description
Is Registered	true if registered using built-in registrar or cache
Is Not Registered	true if not registered at all
Is Registered Locally	true if registered using built-in registrar using the action "Save REGISTER contact"
Is Not Registered Locally	true if not registered using built-in registrar
Is Registered Remotely	true if URI cached using "Enable REGISTER caching"
Is Not Registered Remotely	true if URI not cached

Supported operators for "Date and Time" are:

Table 5: "Date and Time" Operators

Operator	Description
is after now plus	The left operand is checked for being after the current time plus an offset
is before now plus	The left operand is checked for whether it is before now plus an offset
is after now minus	The left operand is checked whether it is after the current time minus an offset
is before now minus	The left operand is checked for whether it is before the current time minus an offset

Note that the offset is optional, and it is always added or subtracted to the current time before the comparison.

Available operators for "Supported header" and "Require header" conditions are:

Table 6: "Date and Time" and "Require header" Conditions

Condition	Description
contains	Checks for presence of given option tag in Supported or Require header field
does not contain	Checks for absence of given option tag in Supported or Require header field

Supported operators for "Request source" are:

Table 7: "Request source" Operators

Operator	Description
is	Matches if request being currently processed was generated in accordance with right operand.
is not	Matches if request being currently processed was NOT generated in accordance with right operand.

Supported right operands for "Request source" are:

Table 8: "Request source" right side operands

Operand	Description
local	request locally generated by SBC
call transfer	request locally generated by SBC as result of unattended call transfer

6.2.3 Condition Values and Regular Expressions

Values in a condition may be of several kinds. They are interpreted in the following descending order.

- |*SBC*| *Escape Codes*. These are characters prefixed by backslash (\) that are supposed to be interpreted literally. These are normally used only for special characters. For example, \\ stands for backslash and \$ stands for the dollar character.
- |*SBC*| *Replacements*. These are variables that refer to different parts of SIP messages or internal variables. They are referred to by $ character followed by variable name and replaced with value of the variable. The variables that can be used are listed in Section *Using Replacements in Rules*.
- *regular expressions*. Regular expressions are expected if one of the regular-expression matching operators is used. ABC SBC uses the "extended POSIX regular expression" syntax. That means, among others, that a section enclosed in parenthesis can be referred to from back referencing expressions in actions' parameters (see Section *Using Regular Expression Backreferences in Rules*), the special characters * (star: zero to any), + (plus: one to any), ? (question mark: none or one), and {a,b} (curly brackets: from a to b) specify the number of occurrences, . (dot) stands for wildcard, ^ (caret) stands for beginning of a string, $ (dollar) stands for end of a string, | (pipe) stands for alternation and square brackets are used for character sets (^ as leading character means negation).
- *literals*. This is the simplest case: a value is used for condition "as is" without further interpretation. For example, in condition "R-URI User == foo", the word foo is matched against the value of userpart of request URI.

Note that this interpretation order determines the condition result. If a regular-expression includes the "end-of-string" character, $, it must be preceded by backslash. Otherwise it will be interpreted in the previous step as an attempt to use a replacement. For example, the "empty string" regular expression must be denoted as "^\$". Another more tricky example is "telephone numbers consisting of a star and two four-digit number blocks". To make sure that a regular expression matches the whole userpart of a URI and not just a part of it, it must begin with "^" and end with "$". Because star has a special meaning in regular expression language, it must be preceded with a backslash. And because the backslash may have special meaning in the ABC SBC GUI, it must appear twice. The resulting expression looks like this

```
^\\*([0-9]{4,4})([0-9]{4,4})\$
```

Also note that an expression in the right operand can contain replacements, but can not contain back-references as described in Section *Using Regular Expression Backreferences in Rules*. These are only available as action parameters.

6.2.4 Actions

Actions define how a request shall be treated. There are many kinds of, described in the following sections of this guide as well as in the Section *Reference of Actions*.

The key functionality available through the actions covers the following aspects of VoIP processing:

Table 9: Actions

Action Group	Purpose
SIP Mediation	Manipulation of identity and URIs, header fields, and response codes. See Section *SIP Mediation*.
SDP Mediation	Manipulation of codec and early media negotiation. See Section *SDP Mediation*.
Management and Monitoring	Logging traffic and reporting SNMP statistics. See Section *Diagnostics Dashboard* and *Using SNMP for Measurements and Monitoring*.
Traffic Shaping	Putting quota on SIP and RTP traffic and reporting violations. See Section *Traffic Limiting and Shaping*.
Media Processing	Handling RTP traffic: RTP anchoring, RTP/SRTP conversion, RTP inactivity detection, audio recording and transcoding. See Section *Media Handling*.
Identity	Verifying a 2FA PIN number via DTMF and enrolling a user for 2FA number verification.
SIP dropping	Eliminating non-compliant traffic, silently or with a SIP response. See Section *Manual SIP Traffic Blocking*.
Scripting	Processing of internal variables that are used to link multiple actions together using intermediate results stored in variables. See Section *Binding Rules together with Call Variables*.
Register Processing	REGISTER caching and uncaching, registrar, throttling. See Section *Registration Caching and Handling*.
External Interaction	Queries to external servers by REST or ENUM or internal pre-provisioned database. See section *Advanced Use Cases with Provisioned Data*.
NAT Handling	Fixing SIP to facilitate NAT traversal in a safer way than by the SIP specification. See Section *NAT Traversal*.
Other	Some other actions.

6.3 Using Replacements in Rules

In many cases, the conditions values and parameters of actions are not known in advance: they depend on elements of processed SIP messages and results of the message processing. Therefore, it is possible to compose the parameters of special strings that refer to SIP processing status. These strings are called "replacements" and are denoted by a dollar ("$") sign followed by an identifier. Each instance of a replacement is replaced by its value when the rule is evaluated.

For example, **$aU** is a replacement for the User part of the *P-Asserted-Identity* header; **$th** is a replacement for the host part of the *To* header. The action Set R-URI with the parameter set to **sip:$aU@$th** combines mentioned parts of *P-Asserted-Identity* and *To* headers of the incoming request and puts them into the request URI of the outgoing request.

All supported replacements are listed in the table below.

Note that these special characters should be backslash-escaped as follows:

- \ → \\
- $ → \$

It is important to know that if a mediation action (Section *SIP Mediation*) changes content of SIP message, the substitution expression will refer to the value **after** modification. E.g., if you apply the rule action "SetFrom(sip:new@from.com)", $fu will return new@from.com!

** Repl. group **	** Replacements**	** Description**
$r	$r.	request-URI; note that the expression refers to current request URI which may be changed during the course of request processing
	$ru	user@host[:port] part of request URI
	$rU	R-URI User
	$rd	R-URI Domain (host:port)
	$rh	R-URI Host
	$rp	R-URI Port
	$rP	R-URI Parameters
$f	$f.	From header
	$fu	user@host[:port] part of From URI
	$fU	From User
	$fd	From Domain (host:port)
	$fh	From Host
	$fp	From Port
	$fn	From Display name
	$fP	From Parameters
	$ft	From Tag
	$fH	From header Headers
$t	$t.	To header
	$tu	user@host[:port] part of To URI
	$tU	To User
	$td	To Domain (host:port)
	$th	To Host
	$tp	To Port
	$tn	To Display name
	$tP	To Parameters
	$tt	To Tag
	$tH	To header Headers
$a	$a.	P-Asserted-Identity header
	$au	user@host[:port] part of P-Asserted-Identity URI
	$aU	P-Asserted-Identity User
	$ad	P-Asserted-Identity Domain (host:port)
	$ah	P-Asserted-Identity Host
	$ap	P-Asserted-Identity Port
	$aP	P-Asserted-Identity Parameters
	$aH	P-Asserted-Identity Headers
$p	$p.	P-Preferred-Identity header
	$pu	user@host[:port] part of P-Preferred-Identity URI
	$pU	P-Preferred-Identity User
	$pd	P-Preferred-Identity Domain (host:port)
	$ph	P-Preferred-Identity Host
	$pp	P-Preferred-Identity Port
	$pP	P-Preferred-Identity Parameters
	$pH	P-Preferred-Identity Headers
$c	$ci	Call-ID
$C	$C.	complete Contact-HF

continues on next page

Table 10 – continued from previous page

** Repl. group **	** Replacements**	** Description**
	$Ci	user@host[:port], port is included if present in Contact-HF
	$Cx	x' is anything supported for other URIs
$s	$si	Source (remote) IP address
	$sp	Source (remote) port number
$d	$di	expected destination host
	$dp	expected destination port
$R	$Ri	Destination (local/received) IP address
$R	$RI	Destination IP address – like above but when a public IP is configured on the receiving interface, its value is used instead.
	$Rp	Destination (local/received) port number
	$Rf	local/received interface id (0=default)
	$Rn	local/received interface name (SBC interface name)
	$RI	local/received interface public IP
	$Rt	local/received transport protocol one of: tcp, tls, udp, ws (WebSocket), wss (secure WebSocket)
$H	$H(headername)	value of header with the name headername; not applicable to from/to/ruri/contact for which specific replacements must be used
	$HU(headername)	header headername (as URI) User
	$Hd(headername)	header headername (as URI) domain (host:port)
	$Hu(headername)	header headername (as URI) URI
	$Hd(headername)	header headername (as URI) domain (host:port)
	$Hh(headername)	header headername (as URI) host
	$Hp(headername)	header headername (as URI) port
	$Hn(headername)	header headername (as URI) display name
	$Hp(headername)	header headername (as URI) parameters
	$HH(headername)	header headername (as URI) headers
$m	$m	request method
$V	$V(gui.varname)	value of Call Variable varname
$B	$B(cnum.rnum)	**value of backreference with** *rnum* number from the condition with *cnum* number
$U	$Ua	register cache: originating AoR
	$UA	register cache: originating alias
$_	$_u(value)	value to uppercase
	$_l(value)	value to lowercase
	$_s(value)	length of value (size)

continues on next page

Table 10 – continued from previous page

** Repl. group **	** Replacements**	** Description**
	$_5(value)	MD5 of value
	$_r(value)	random number 0..value, e.g. $_r(5) gives 0, 1, 2, 3 or 4
$#	$#(value)	value URL-encoded
$time	$time(value)	time format as described in the libc strftime() function. ie: $time(%m-%d-%y-%H-%M)
$attr	$attr(value)	value of the given global attribute
$cntr	$cntr(value)	value of the given counter defined by *User Defined Counters*
		e164 support
$e164	$e164(number, country_code)	Convert the number parameters to the e164 format of the country code. ie: $e164(0635215099, FR) = +33635215099
$T	$T(number, country_code)	Return the number type given the country code. The value are the same as the *libphonenumber short-url.at/iwxyJ*.
$rc2cc	$rc2cc(region_code)	Return the country code of the region. ie: $rc2cc(FR) = 33
$cc2rc	$cc2rc(country_code)	Return the region code of the country. ie: $rc2cc(33) = FR

6.3.1 Example Use of Replacement Expressions

In the following example, see Fig. *Using Replacements*, we set up the outgoing INVITE request as follows:

- set Request URI of the outgoing INVITE request to the user part of the *P-Asserted-Identity* header ($aU) combined with the host part of the *To* header ($th) of the incoming INVITE request
- set host part of the *To* header to the value of the *P-NextHop-IP* header ($H(P-NextHop-IP)) of the incoming INVITE request (the user part will not be changed)
- convert the user part and the host part of the *From* header into lower case (<sip:\protect\T1\textdollar_l(\protect\T1\textdollarfU)@_l(\protect\T1\textdollarfh)>).

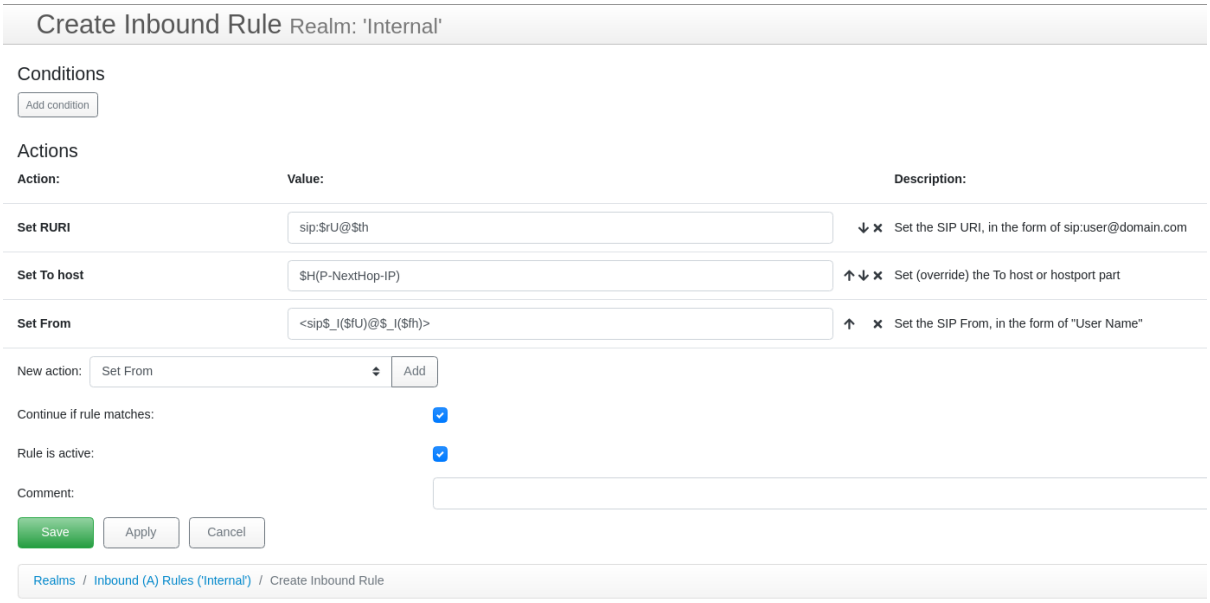


Fig. 3: Using Replacements

The effects of this transformation on a SIP message is depicted in Fig. *Effects of using replacements*:

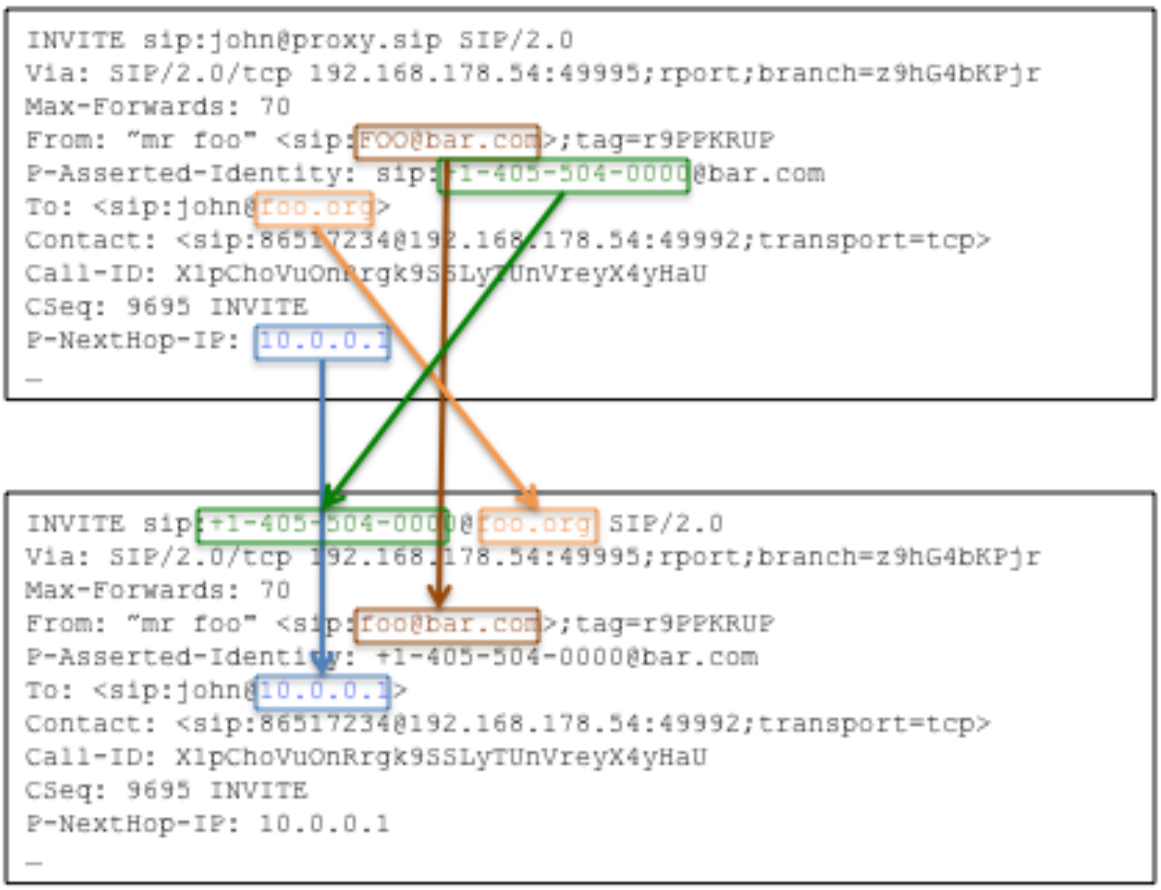


Fig. 4: Effects of using replacements

6.4 Using Regular Expression Backreferences in Rules

Whenever a regular expression match is executed in a rule condition, the matched substrings can be later used in subsequent actions or conditions. The matched result is referred to by so called "backreferences".

Backreferences are used by the replacement **$B(c.r)**. The first index in the backreference, **c**, denotes the index of the condition, where the first condition has the index 1, the second condition the index 2, and so forth. The second index, **r**, denotes the index of the substring selection in the regular expression, where the first selection has the index 1, the second the index 2, an so forth.

In the following example, see Fig. *Using backreferences*, we use backreferences to separate protocol discriminator ("sip" or "tel") from the rest of request URI. These two parts are matched in the regular expression in the 2nd condition and are therefore referred to as $B(2.1) and $B(2.2). Particularly, the example saves the protocol discriminator from the request URI in an INVITE request to a call variable called **uri_scheme**. Further it enforces the "sip" scheme for the R-URI of the outgoing INVITE request.

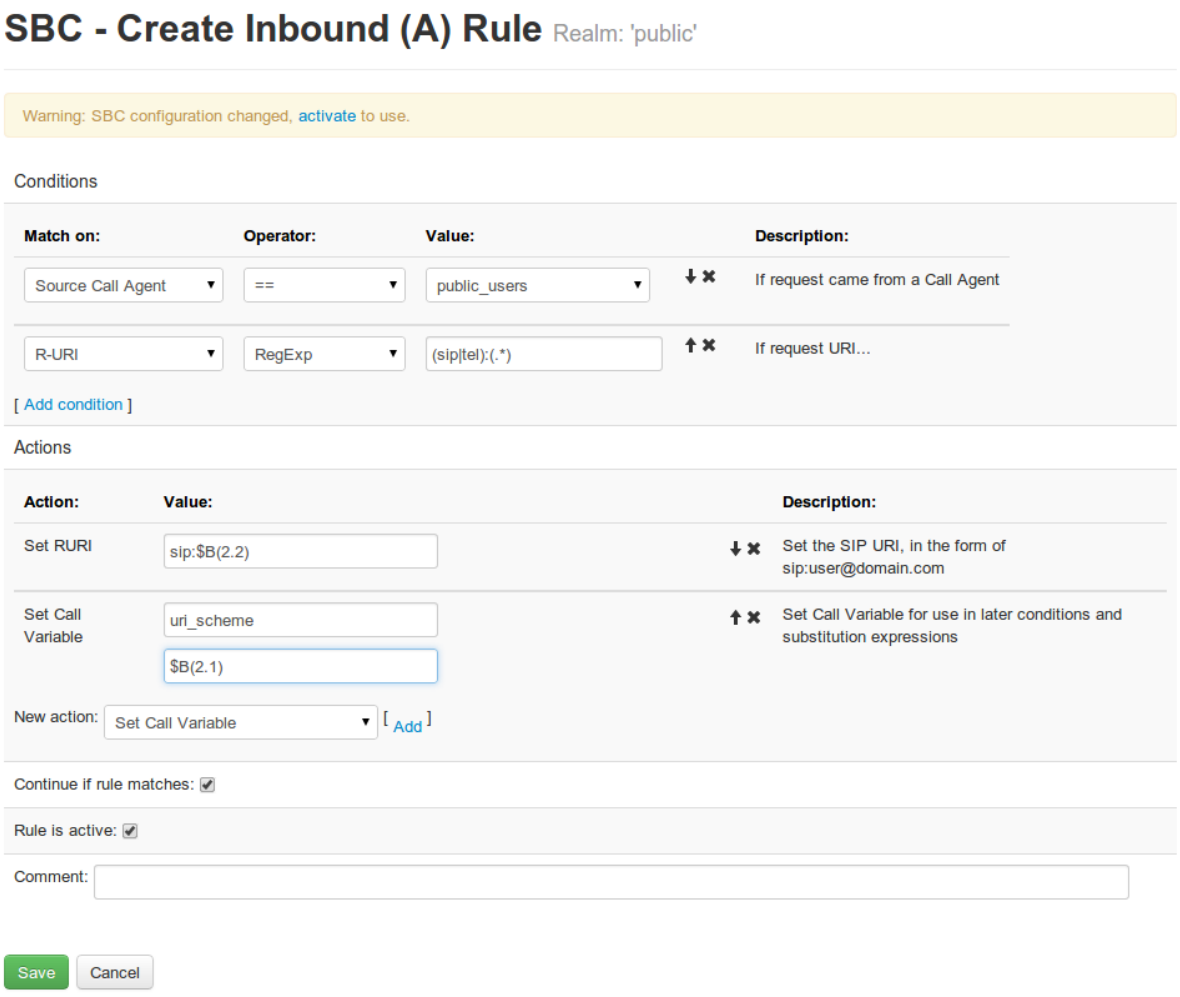


Fig. 5: Using backreferences

6.5 Binding Rules together with Call Variables

Call Variables are a very powerful tool in the ABC SBC, because they can bind together different rules or rule sets. Call Variables can be set by rules to any value using the **Set Call Variable** action. This variable will the persist during the lifetime of the call. They can be set to a different value by a subsequent rule, again with the **Set Call Variable** action.

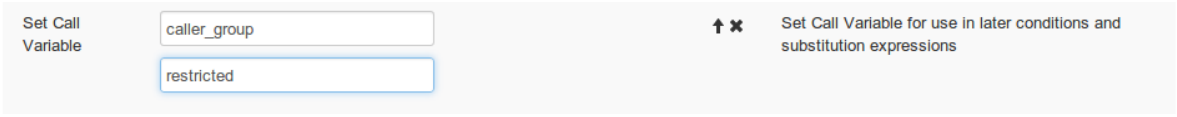


Fig. 6: Setting Call Variables

Values of Call variables can be tested with the **Call Variable** condition using several operators: ==, !=, "RegExp", "does not match RegExp", "begins with" and "doesn't begin with". Operands may be literal strings, regular expression if the "Regexp" operators are used, and they may contain Replacements (see Section *Using Replacements in Rules*).

An additional condition, "Call Variable Existence", allows to test if a variable exists and accurately discriminate it from the case when it is empty-valued. This may be particularly handy when table lookups are used as described later in Section *Provisioned Tables*. Otherwise reference to undefined variables always returns empty string.

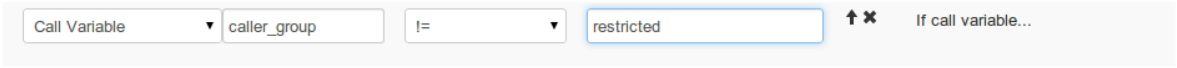


Fig. 7: Testing Call Variables

They can also be used in other actions using the replacement expression $V(gui.varname), where *varname* refers to the name of the variable, e.g. $V(gui.caller_group).

Fig. 8: Using Call Variables

The following example shows how a variable is assigned a value using the "Set Call Variable" action (see Figure *Example for using Call Variables*), tested for a specific value "restricted" (see Figure *Testing Call Variables*) and referred to from an action for adding a new *Reason* Header field using the $V replacement (see Figure *Using Call Variables*).

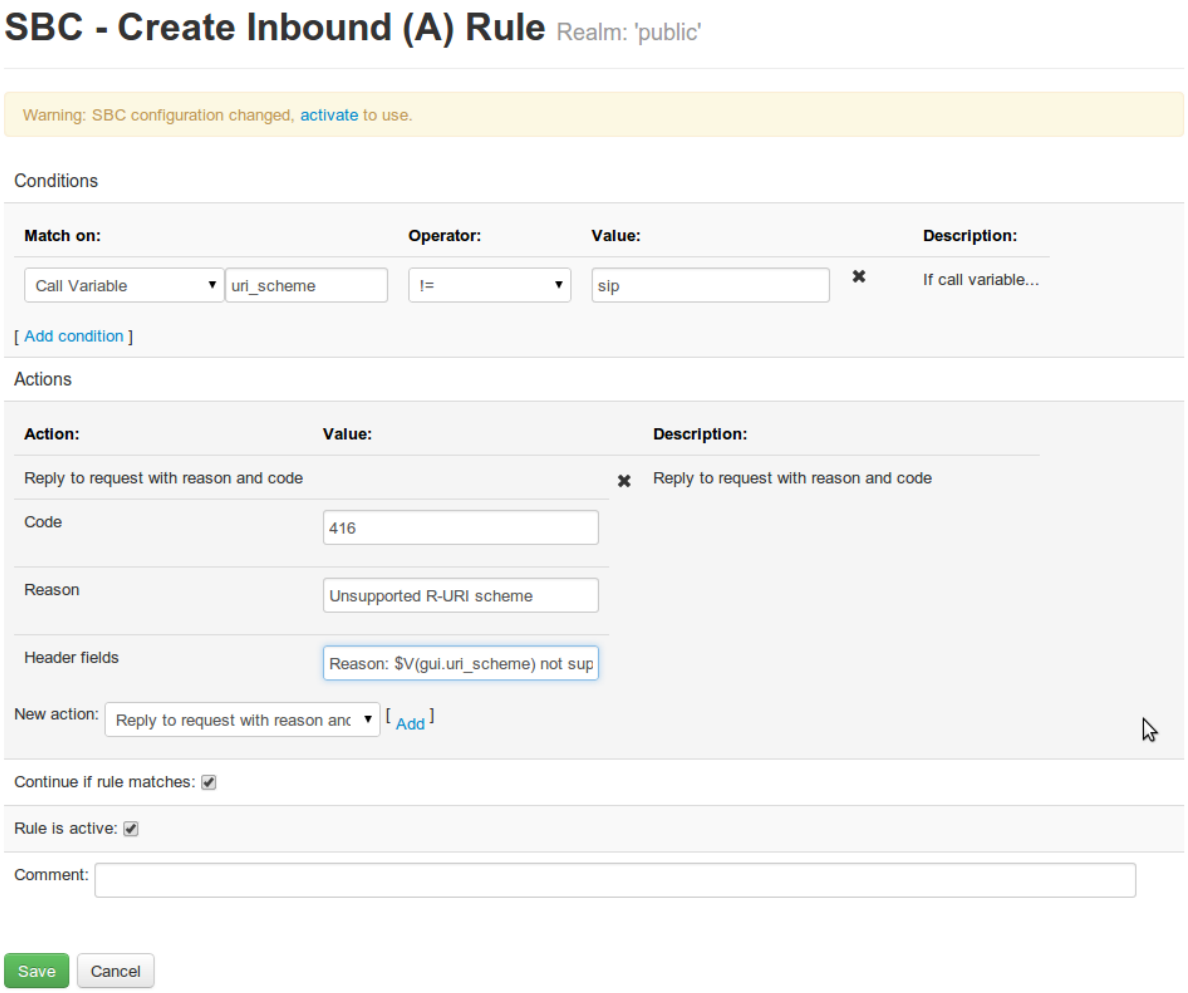


Fig. 9: Example for using Call Variables

6.6 SIP Routing

The key functionality of the ABC SBC is that of SIP routing: based on criteria chosen by the administrator, the SIP destination for a SIP dialog is chosen. The routing decision is the step "B" in the A-B-C process: After A-rules are applied based on who sent the SIP traffic to the ABC SBC, the destination Call Agent is chosen in the B-rules. The final step is processing of the outbound C-rules, that are specific to the Call Agent chosen in the step B.

The routing decision is also an important part of the network reliability concept: it has implications to the way how traffic is re-routed if downstream destinations are unavailable or overloaded.

Unlike steps A and C, the routing step B is global: it is executed for every combination of inbound and outbound call agents and realms. It can be seen like the wiring board between these. Also unlike A and B rules, matched rules can have only one action: selection of the destination.

The routing rules are processed sequentially until one is found that matches. Repetitive rules, such as least-cost-routing tables, can also be managed by provisioned tables as described in the Section *Provisioned Tables*. If no route matches, the ABC SBC stops processing the SIP request and returns a 404 SIP response.

The outcome of the routing process is unique determination of the destination Call Agent. This decision determines the following aspects:

- which C-rules are executed,
- which backup Call-Agent is used, if forwarding to the chosen Call Agent fails,

- which interfaces are used for forwarding,
- which IP address or addresses are used as next hop for forwarding.

Note that the routing process is applied only to dialog-initiating or out-of-dialog SIP requests. During the dialog life-time, routing of in-dialog SIP requests follows a fixed path established in the process of the dialog initiation with the peering SIP devices. The path may or may not be the same as that of dialog-initiating transaction and is formed using Record-Route and Contact header-fields as governed by the **RFC 3261** specification. Only if the "use on first request only" option is turned off, or "Dialog NAT handling" is enabled, the ABC SBC routes subsequent requests in a sticky way to the same hop as the initial one.

The following subsections describe how routing rules are organized and how to use the three types of routing rules: "static" for well-known next-hop Call Agents, "table-based dynamic" for a massive amount of static routes, and "request-URI based" for destinations identified in request URI. Eventually we show how the abstract destination is translated into next-hop IP addresses for request forwarding.

6.6.1 Routing Rules (B)

The routing rules are an ordered list of routes, which are processed one by one after completion of A rules processing. When the first rule condition matches, the destination call agent is chosen and route processing stops.

The configured Routing (B) rules can be viewed when clicking on the "Routing" menu entry.

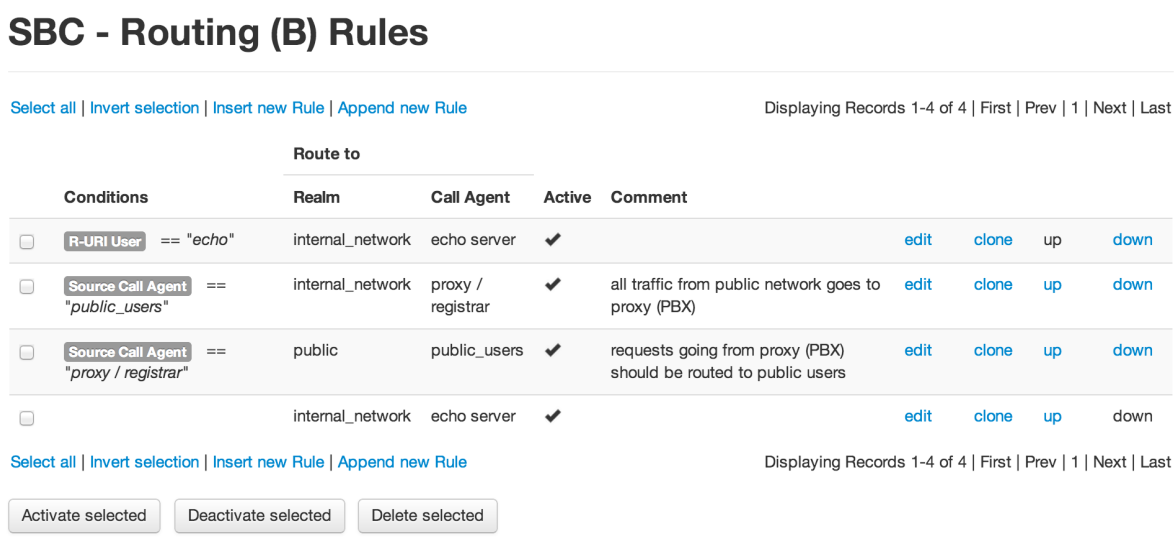


Fig. 10: List or routing rules

A new routing rule can be created either at the beginning of the list ("Insert new Rule") or at the end ("Append new Rule").

When creating a new Routing (B) rule, one or more conditions can be entered, see Fig. *Define matching conditions for routing*, similar to the other types of rules (inbound (A) and outbound (C) rules).

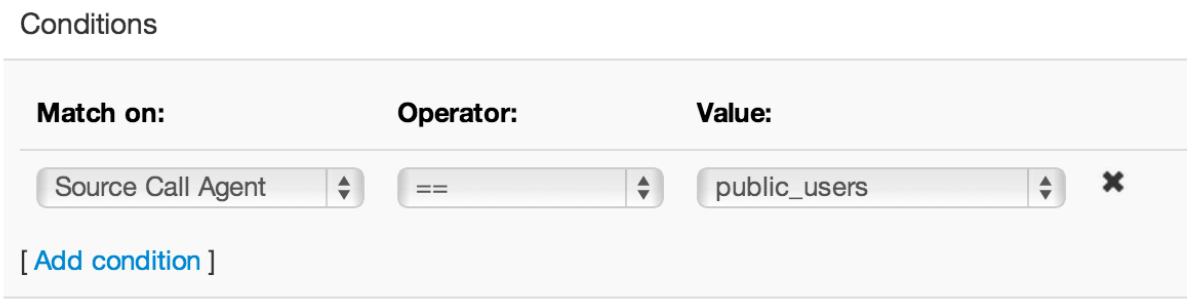


Fig. 11: Define matching conditions for routing

It is also possible to define a default routing rule by omitting the condition(s). In this case, the rule will always match, and thus finish the Routing rules evaluation. In case there is no such default rule and no other rule matches, the ABC SBC answers the request with a *404* response.

There are several types of routing rules, that can be combined with each other and processed in sequential order:

- **Static route** – In a static route, all data describing the routing decision must be entered manually. If the static route matches, routing finishes, otherwise it proceeds to the next rule.
- **Dynamic route** – if multiple repetitive rules, such as with Least Cost Routing, are configured, they are better placed in a provisioned table as described in the Section *Provisioned Tables*. To make a lookup in the table, select the table name in the "Route using" drop down list and indicate by what key the lookup shall be performed. Like with static routes, if a matching entry is found, routing finishes, otherwise it proceeds to the next rule.
- **Call Agent based on R-URI** – the ABC SBC tries to find a Call Agent that matches host in request URI. This can be particularly useful if A-rules change hostname in request URI, for example by ENUM lookup. If the lookup yields an address of a valid Call Agent, it is used for routing and routing finishes, otherwise it proceeds to the next rule.

These route types are described in more detail in the following sections.

Validity of routing rule can be limited to some nodes only. In this case the routing rule is not executed on other nodes. Routing rule can be assigned to only those nodes which belongs to any of the config groups assigned to the particular routing table.

6.6.2 Static Routes

Static route is the simplest type of routing rule: the administrator explicitly chooses the destination Call Agent. If no further specific treatment is desired, that's all. The Call Agent is chosen, subsequently its C-rules are executed and eventually signaling is forwarded through the interface associated with the Call Agent. This routing method is applicable to all Call Agent types but those identified by a subnet address – these are used primarily for matching of incoming traffic and do not uniquely identify a destination forwarding address.

The choice of Call Agent is accompanied by several other options. The most important is that of routing method which specifies how the next-hop IP address is determined. Either it is determined from request URI or from pre-provisioned information. Note that whichever method is chosen to determine the next-hop IP address, Call-Agent does not change and its C-rules are used for request processing. Both methods may yield multiple IP addresses, in which case the ABC SBC load-balances among them by their respective priorities.

The "Route via R-URI" method uses the request URI to find out the next-hop IP address. That is particularly useful when A-rules altered the request URI using actions like reverse registration cache or ENUM lookup. If the host part of request URI includes a DNS name that resolves to multiple destinations per **RFC 3263**, the ABC SBC load-balances among the respective destinations by their priorities.

If "Set Next Hop" (also known as "outbound proxy") is used instead, the next-hop IP address is determined using pre-provisioned information. Either the IP address (or addresses) associated with the Call Agent is taken, or these are explicitly overridden using the option "Use another destination instead of CAs' destination(s)". Please note

that, if "Use another destination instead of CAs' destination(s)" is used, backup CA will not fully work (C rules from the CA **will** be applied to outgoing request). In that case, if the first CA fail, all calls are sent to the same IP, located in "Use another destination instead". Further method-specific options include:

- "Use on first request only" – this option changes default behavior for forwarding subsequent in-dialog requests. By default when turned off, all subsequent outbound requests will follow exactly the same the path of the previous dialog-initiating request. If however this option is turned on, the next-hop logic for subsequent requests is governed only by the SIP standard procedures. Particularly, if the next hop in the INVITE path was a non-record-routing proxy, it will not be included in request's path.
- "Update R-URI Host". This option rewrites host part of request URI with the address of the next hop. By default it is turned off and the request URI remains untouched when forwarding.
- "Add Route HF". This option is also known as "preloaded Route". It prints the next-hop destination in Route Header-field. Use only if downstream SIP hop is known to require such behavior.

The following advanced options can be also used with both methods:

- "Replace DNS name in R-URI through the resolved IP address" makes sure that if DNS names appears in request URI, it is rewritten to IP address before forwarding.
- "Force transport". Allows to override transport protocol to be used for the next hop. One of the following protocols can be chosen: UDP, TCP, TLS.
- "Enable redirect handling". If this option is turned on, incoming 302 are not passed upstream. Instead, the ABC SBC takes content of Contact header field and uses it as another next-hop for forwarding the original request. Particularly the Contact URI in the 302 response is used to rewrite request URI, determine the next-hop IP address and look up a Call Agent whose C-rules are processed. Note that an error occurs and a 500 response is sent upstream if none or more than one matching Call-Agents are found, or the Contacts include DNS names. Use this option with care only for trusted destinations since the 3xx responses may greatly impact where and how requests are forwarded.

An example is shown in Figure *Static Routing destination*: the Call Agent "users" is chosen so that its C-rules will be processed. There is no additional IP address included in the "set-next-hop" choice of routing, so that the dialog-initiating request is forwarded to IP addresses associated with the particular Call Agent.

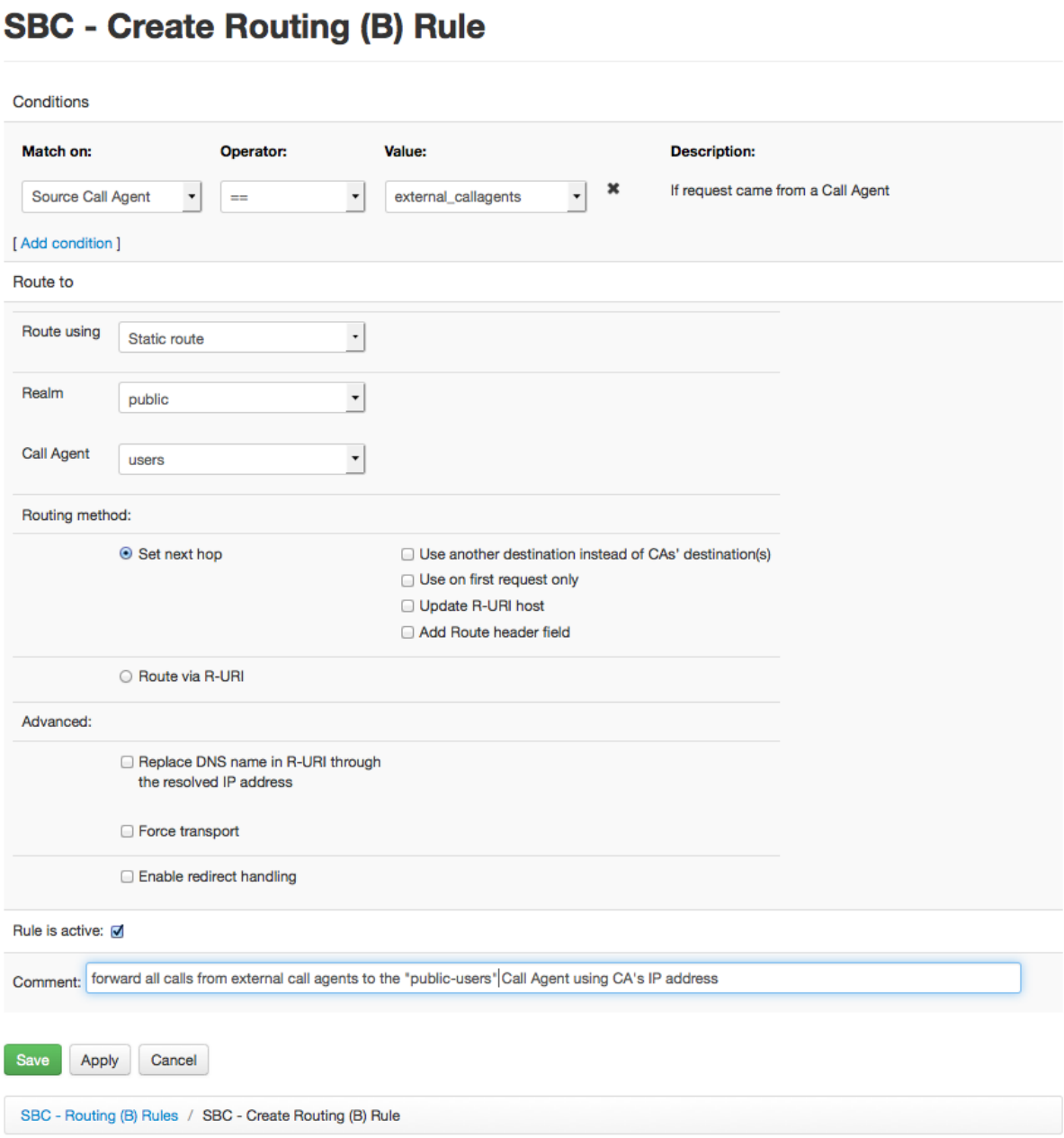


Fig. 12: Static Routing destination

6.6.3 Table-based Dynamic Routes

Long repetitive routing rule sets can be better managed as tables. All other aspects of the routing logic remain the same as with statically defined rules.

To deploy dynamic rules the following steps must be performed:

- definition of a routing table (see Section *Configuring Tables*)
- definition of routing lookup performed against the table in B-rules (see example in Figure *Configure a route lookup in a provisioned routing table*)
- filling in the routing table with routing data (see example in Figure *Adding a new entry to routing table*)

The lookup definition requires two parameters: name of the table defined in the first step, and the value used to lookup a matching table row defined in the second step. The value is defined in form of a *replacement expression*. For example, $rU can be used to trigger lookups by user part of request URI.

The table than includes rows identified by unique keys. In the screenshots bellow, the user part of request URI ($rU) is compared against a row with key 911. If a match occurs, the call agent "external_callagents" is used for call forwarding.

When the table lookup is performed and the value matches no key, routing proceeds to the next entry in the routing table. If there is no more such, routing fails and the SIP request is declined using the *404* SIP response.

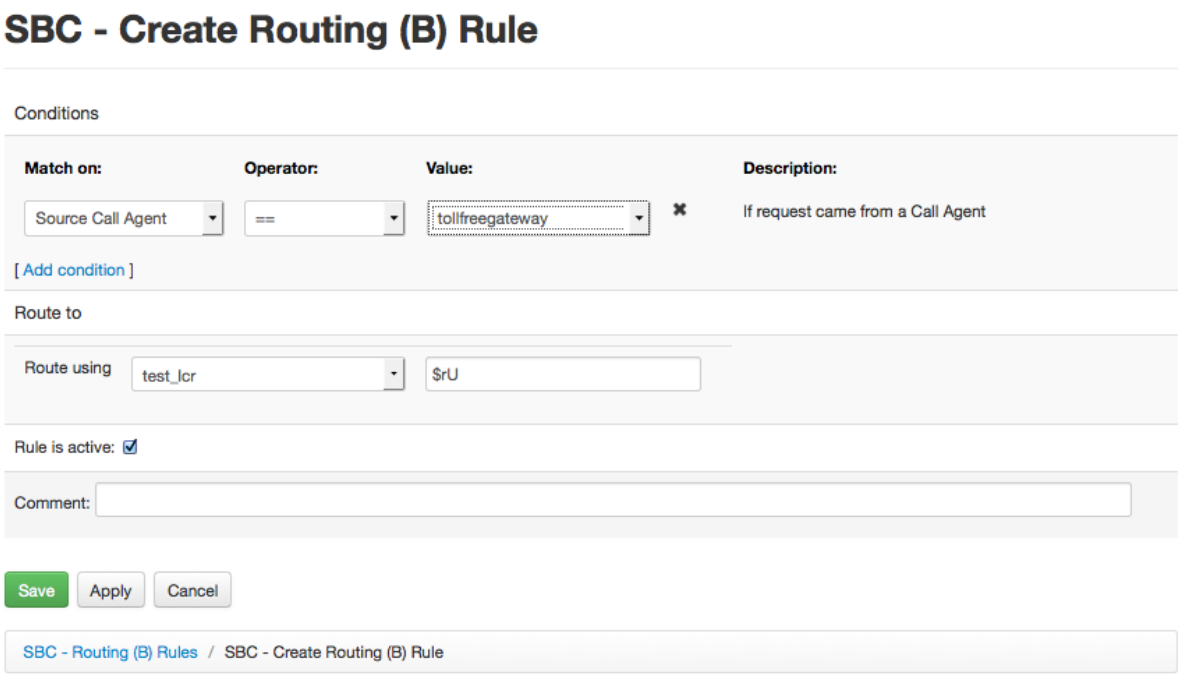


Fig. 13: Configure a route lookup in a provisioned routing table

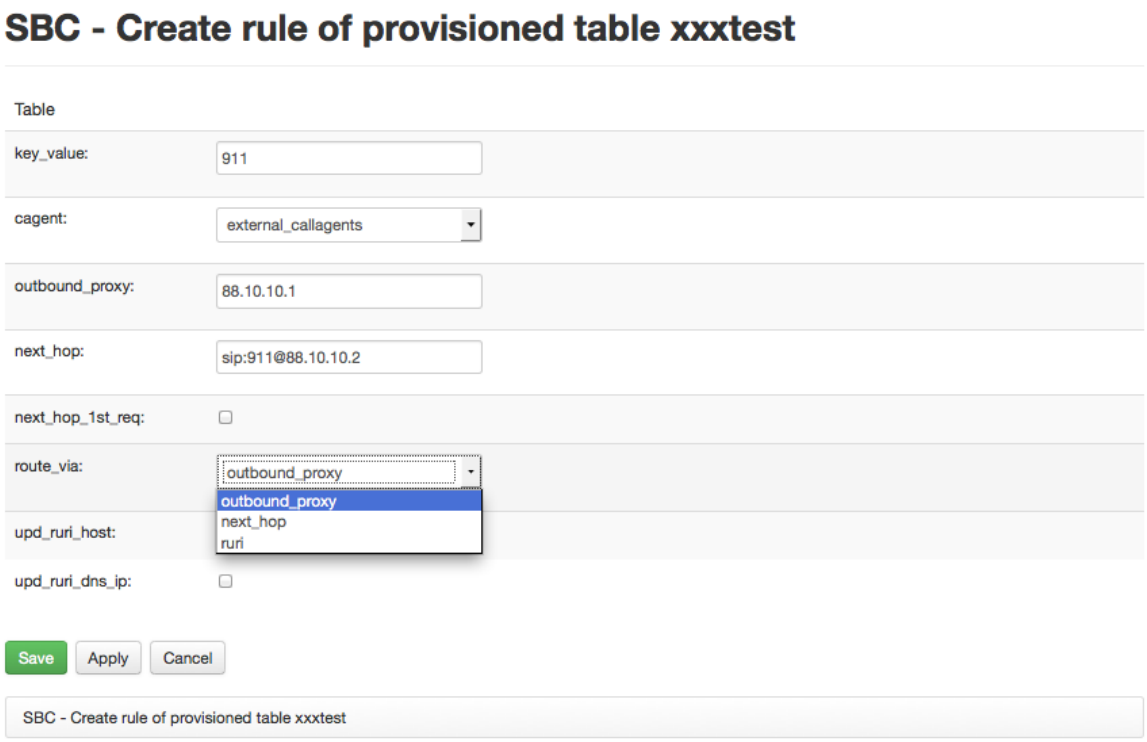


Fig. 14: Adding a new entry to routing table

6.6.4 Request-URI Based Routes

In some scenarios, the next-hop Call Agent is not exactly known at the time of devising a routing policy. Instead it is known that a request URI identifies the Call Agent. This is often the case if the request URI is rewritten by an external query, such as ENUM or REST. There would be little point in formulating rules like "if a CA's IP address present in R-URI, route to the CA" for every single CA.

Therefore there is the "route via R-URI" routing type, which finds a Call Agent based on address in request URI and if found, routes to it.

Note that this is different from the "route via R-URI" option, which is only used to override the transport destination but does not determine the Call Agent with its rules.

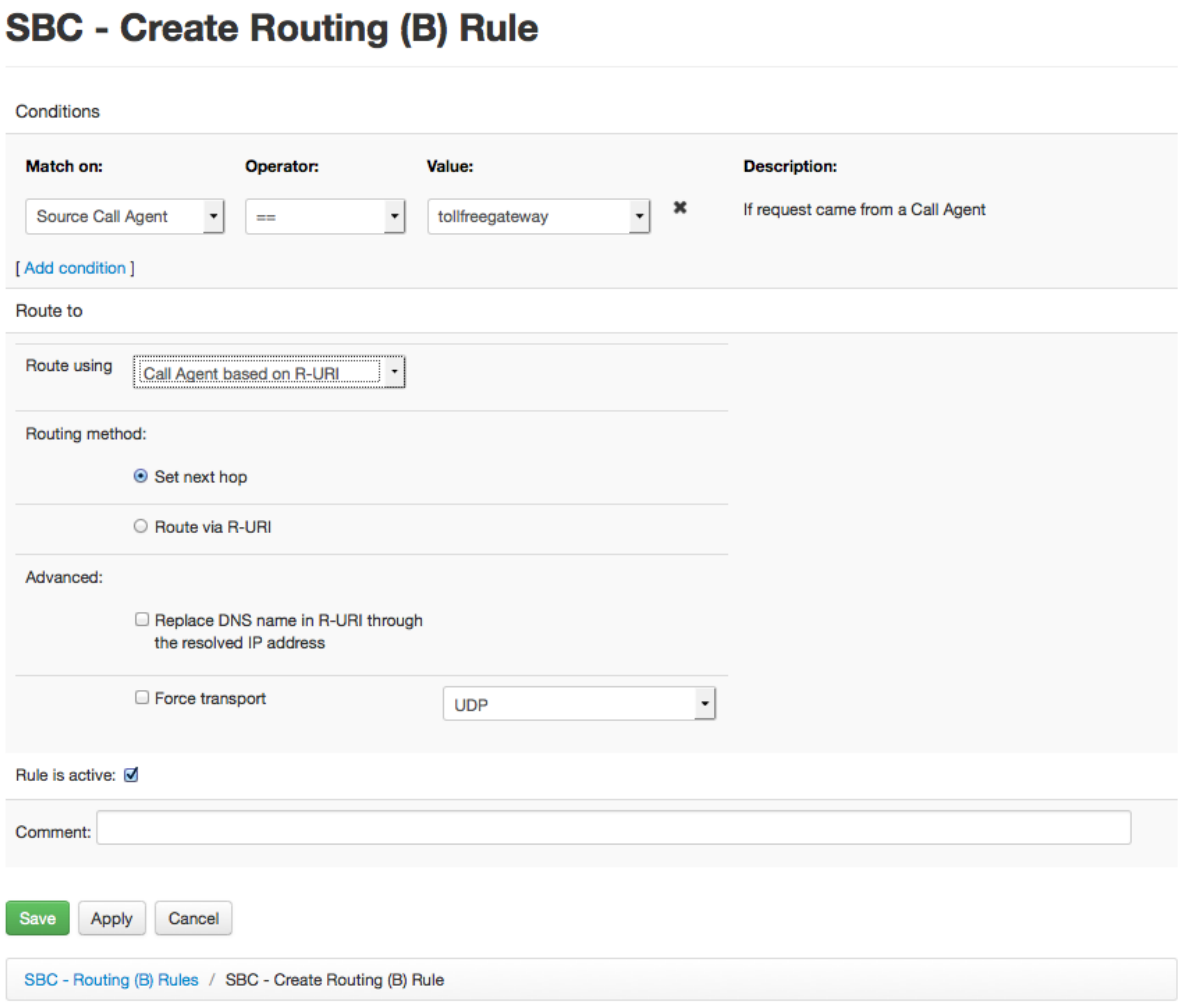


Fig. 15: Route by Request URI

Like with Static Routes, there are two routing methods for determining the next IP-hop: Either it is taken from request-URI (the "Route via R-URI" method) or it is taken from Call Agent's profile. The difference is subtle because by use of the lookup an IP address gained from the request URI must match an IP address of a Call Agent. A difference may occur when some other IP addresses linked with the DNS name are different from those linked with the Call Agent's profile. Also if an IP address comes in request URI and the "route-via-r-uri" method is used, alternate destinations associated with the Call Agent will not be used.

6.6.5 Determination of the IP destination and Next-hop Load-Balancing

When the destination Call Agent is selected, one or multiple IP addresses are chosen for forwarding. These may come from Call Agent definition, explicit addresses in the route or from request URI. Capability to choose more than one IP address is important for load-balancing downstream hosts and for dealing with their unavailability. If there are multiple IP addresses (so called "destination set") the ABC SBC "hunts" through them based on their priorities to find one that is responsive.

The destination set is formed depending on the choice of routing method described in previous sections. It works the same way for static, dynamic and request-URI based types and it can be one of the following:

- if the "Set-next-hop" routing method is chosen without the "Use another destination instead of CAs' destination(s)" option, the addresses specified in Call Agent's profile are used
- if the "Set-next-hop" routing method is chosen with the "Use another destination instead of CAs' destination(s)" option, the addresses specified in this option are used. Addresses associated with the Call Agent are

not used for forwarding.

- if "Route via R-URI" is chosen, the address is taken from the request URI.

If an address in the destination set is a DNS name, it is resolved to IP address(es) using procedures specified in **RFC 3263** before further processing.

If the resulting destination set includes multiple entries they are attempted in successive order. An 8-second timer is used to try up to 4 destinations, so that the hunting attempts complete before standard SIP transaction timeout of 32 seconds. A 503 response makes the ABC SBC to attempt the next destination in the set immediately.

The hunting order is determined by priorities specified in DNS, "Use another destination" option or CA profile. The way priorities are set complies to the **RFC 2782**: the ABC SBC initially contacts hosts with lowest-number priorities. If there are multiple hosts with the same priority they are tried by probability as defined in their weight field. The weight field specifies a relative weight, larger weights are given a higher probability of selection.

When no responsive destination is found, the ABC SBC will check if there is a backup Call Agent defined in the current Call Agent's profile. If so, it will undo previous mediation changes, process backup Call Agent's C-rules and retry the IP calculation process for the backup Call Agent.

The whole process is shown in Figure *Flowchart of Process for Determination of the Next-hop IP Address*.

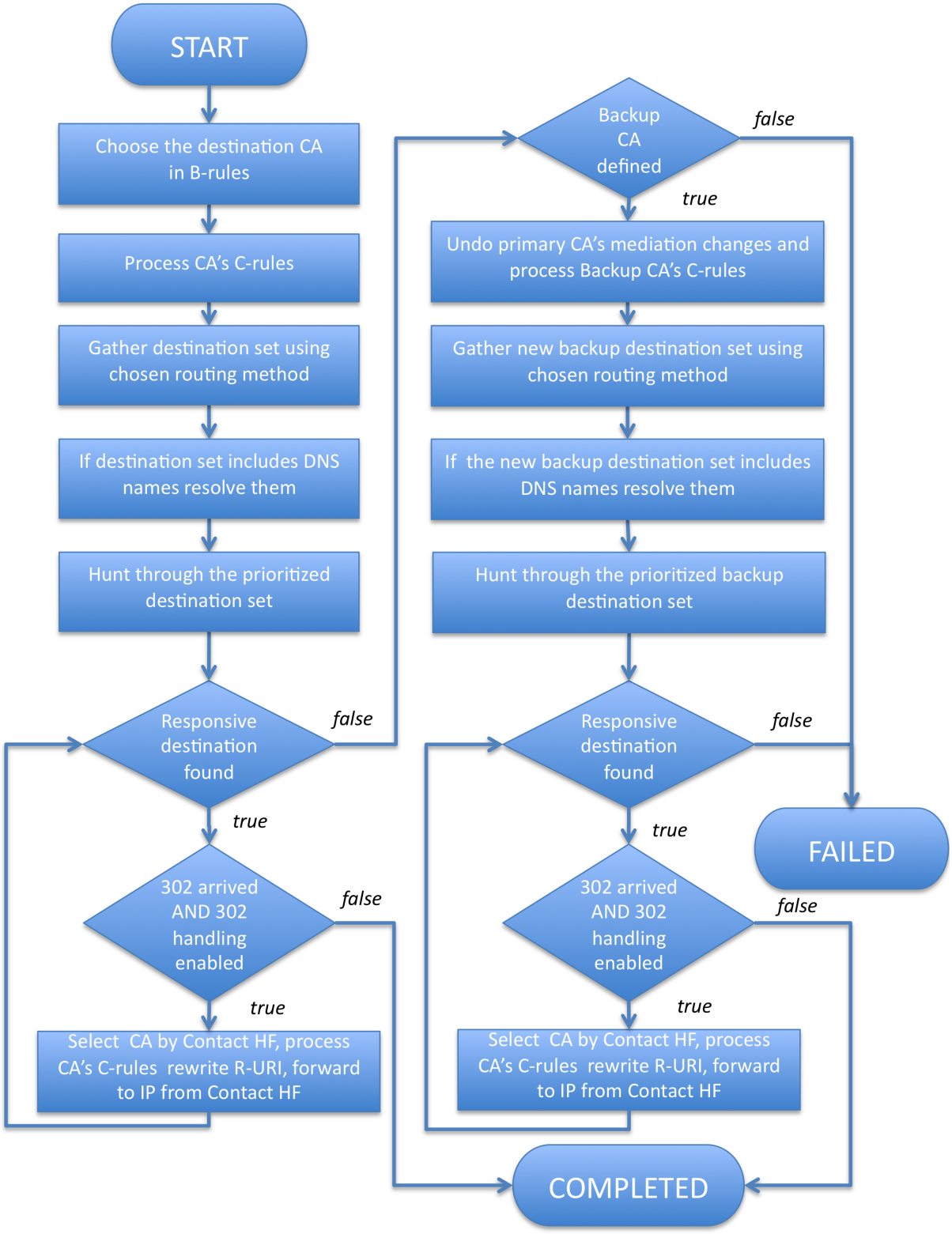


Fig. 16: Flowchart of Process for Determination of the Next-hop IP Address

6.6.6 IP Blacklisting: Adaptive Availability Management

Attempts to forward traffic to IP addresses known to be unavailable would be futile and impair call setup time. Therefore the ABC SBC keeps a "destination blacklist" of IP addresses that were detected as unresponsive. The ABC SBC dispatches no traffic to such destinations until the blacklisting time-to-live expires and the destination is removed from the blacklist.

Additionally the ABC SBC can proactively probe destinations so that their unavailability is detected even before real traffic reaches them. Similarly their renewed availability is detected earlier thanks to the probes even while they are on the blacklist. This is called "Destination Monitor" or "OPTIONS monitoring".

OPTIONS monitoring can be enabled for any SIP Call Agents that are identified by IP addresses. To turn it on, the "Monitoring Interval" under Call Agent's "Destination Monitor" options must be set to a non-zero value. The OPTIONS request are then sent in this interval periodically and have the following form:

```
OPTIONS sip:10.0.0.234 SIP/2.0
Via: SIP/2.0/UDP 10.0.0.155;branch=z9hG4bKo8lw1a70;rport
From: <sip:10.0.0.155:5060>;tag=b280210db5678d3c77dfc06c07acaac3
To: <sip:10.0.0.234>
CSeq: 32603 OPTIONS
Call-ID: 5F1BBCB9-57149447000B9232-0FF75700
Max-Forwards: 0
Content-Length: 0
```

On error, the destination address is placed on blacklist. If it is already there and the OPTIONS transaction completes successfully, the destination address is taken off the blacklist immediately.

Note that this type of blacklisting is different from that used in the context of security policies as described in the section *Manual SIP Traffic Blocking*.

To turn IP blacklisting on, set the time-to-live blacklisting period to a positive value under global options in "Config → Global Config → Default Destination Blacklist TTL" or under Call Agent properties as shown in Figure *Configuration of Destination Blacklisting under Call Agent Properties*.

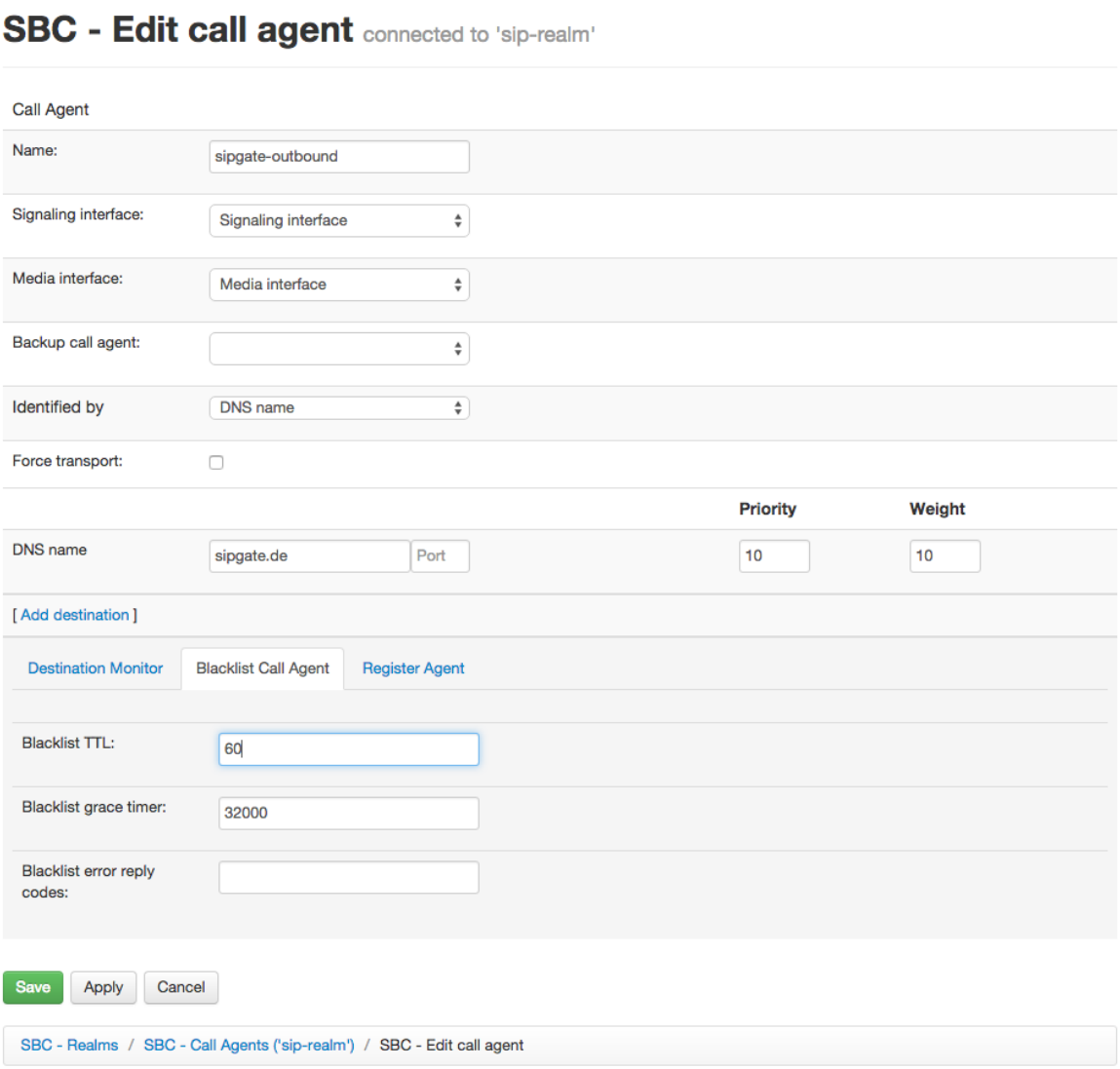


Fig. 17: Configuration of Destination Blacklisting under Call Agent Properties

IP blacklisting occurs in an almost automated way and does typically require minimum administrative attention. Addresses are added to the blacklist once they are identified as unavailable and held on the list for a predefined period of time, known as "time-to-live". The following procedures may still be of use to an administrator:

- If ABC Monitor is used along with the ABC SBC, the history and status of the monitored Call Agents can be tracked in the "Connectivity CA" Dashboard as shown in Figure *Call Agent Availability Lanes*.
- Monitoring blacklisted addresses. It is possible to inspect the addresses which are currently blacklisted. The list is available from the main menu under "Monitoring → Destination Blacklist". (See Section *Destination Blacklists*)
- Manual blacklisting. The administrator may add a new address to the blacklist from the main menu under "Monitoring → Blacklist → New Destination/Save".
- Testing presence on blacklist in rules. Rule conditions may include a test if a Call Agent is present on a blacklist using the "Blacklist" condition type. The condition returns true of all Call Agent's IP addresses are blacklisted.
- Changing Time-to-Live (TTL). The addresses are held on blacklist for period of time specified under "Config → Global Config → Default Destination Blacklist TTL". This value is used for newly blacklisted destinations, unless a CA-specific TTL takes precedence. If TTL is set to zero, no blacklisting takes place.

- Configuring Call Agent specific handling. There are the following options available under Call Agent profile:
  - Destination Blacklist TTL (ms). This value overrides the globally specified time to live.
  - Blacklist grace timer (ms). This value provides some extra time before a downstream element is blacklisted. That is especially useful when there is a proxy server between the ABC SBC and an unresponsive User Agent Server. A too aggressive blacklisting process would otherwise blacklist the proxy before it times out and sends a 408 message and make the proxy and elements behind it unreachable.
  - Blacklist error reply codes. This feature allows to blacklist destinations that answer to monitoring requests using these codes. Useful for example if a destination signals overload condition using 503. To activate the feature, include a comma-separated list of response codes that lead to inclusion of a destination on a black-list.
  - Destination Blacklist for in-dialog requests. This feature allows to blacklist destinations during in-dialog requests. This can be used to allow in-dialog failover to another destination if the currently used destination becomes unavailable during a dialog and thus lands on the destination blacklist.
  - Monitoring interval (seconds). If set to a non-zero value, the ABC SBC tests availability of the destination by sending test message (OPTIONS). This allows to detect unavailable destinations even before a real call hits it. It is recommended to use a value shorter than the blacklisting TTL: if the monitoring period was longer, unresponsive destinations would be considered healthy in the time-window after removing from blacklist before the next monitoring check.

Whenever an address is added to the IP blacklist, an event of type 'notice' is generated. The same occurs when the TTL expires or the destination becomes responsive and the address is removed from the blacklist. The monitoring status is regularly reported to the ABC Monitor using "dest_monit" events.

6.6.7 SIP Routing by Example

One important configuration step is the definition of routing rules, i.e. to which IP address shall incoming traffic be forwarded. If a proper routing entry is not set up, the ABC SBC does not know where to forward traffic and returns the SIP reply "404 Not Found".

In our example, the routing configuration is very simple: What comes from the external Realm is routed to the internal Realm and vice versa. That takes two rules defined from the "Routing" section of the web user interface: Traffic coming from the Call Agent "Proxy" is routed to the Call Agent "Users" in the external realm and traffic coming from the "external" Realm will be routed to the "proxy" Call Agent in internal Realm. The resulting configuration is depicted in Fig. *Example Routing Rules*.

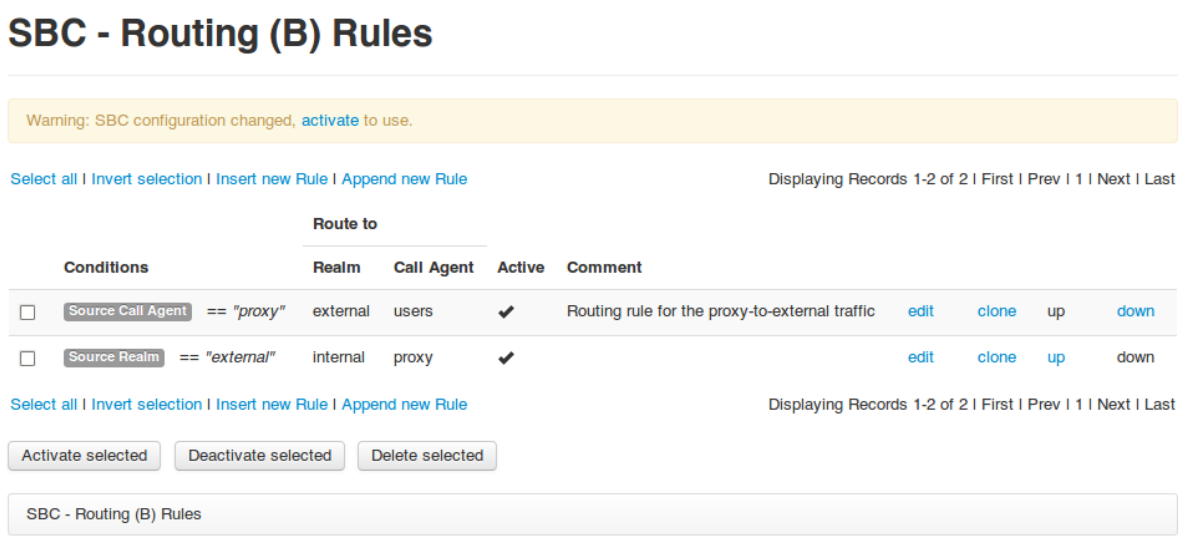


Fig. 18: Example Routing Rules

When you define the respective routing destinations, specifying the abstract Call Agent may not be enough. You may additionally need to help the SBC to determine to which IP address to forward the SIP message and which hostname to use in the Request URI. For example, traffic leaving the SIP proxy carries the final destination in the Request URI. You must configure the ABC SBC to use the IP address from the request-URI as the next hop. The particular configuration is called **Route via R-URI**. On the other hand, all traffic from the public Internet goes to the same proxy server. The appropriate configuration choice is **Set Next Hop**. You may specify the IP address explicitly, if you do not do so, the IP address is taken from definition of the Call Agent.

The routing rule for the proxy-to-external traffic flow is shown in the Figure *Routing Rule for internal to external traffic*, whereas the rule for the opposite direction is shown in the Fig. *Routing Rule for external to internal traffic*. That's it. We now have the routing policy which specifies that traffic from the external Realm shall be forwarded to the internal Realm and vice versa. This simple policy can in fact serve an incredible number of use-cases.

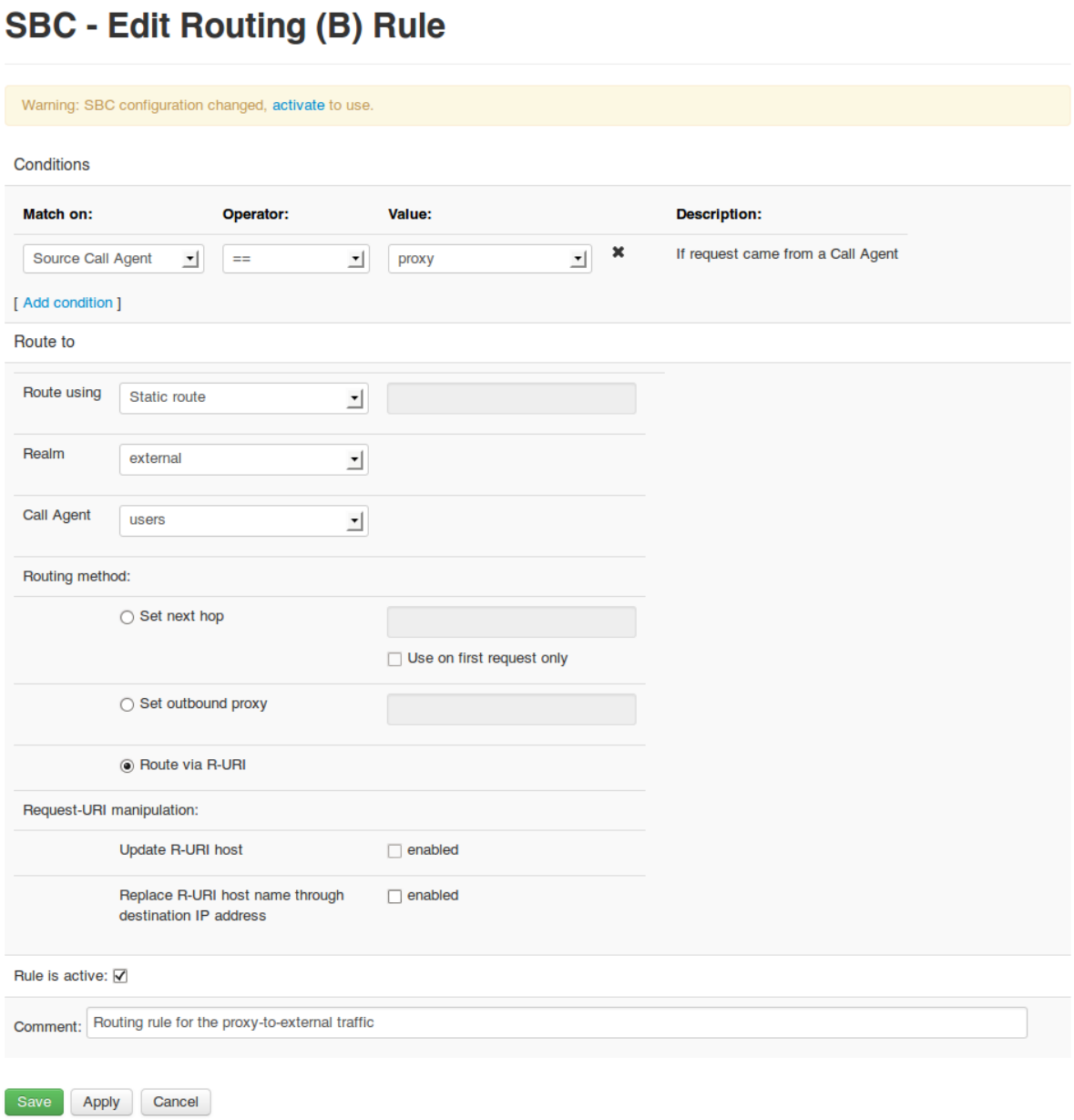


Fig. 19: Routing Rule for internal to external traffic

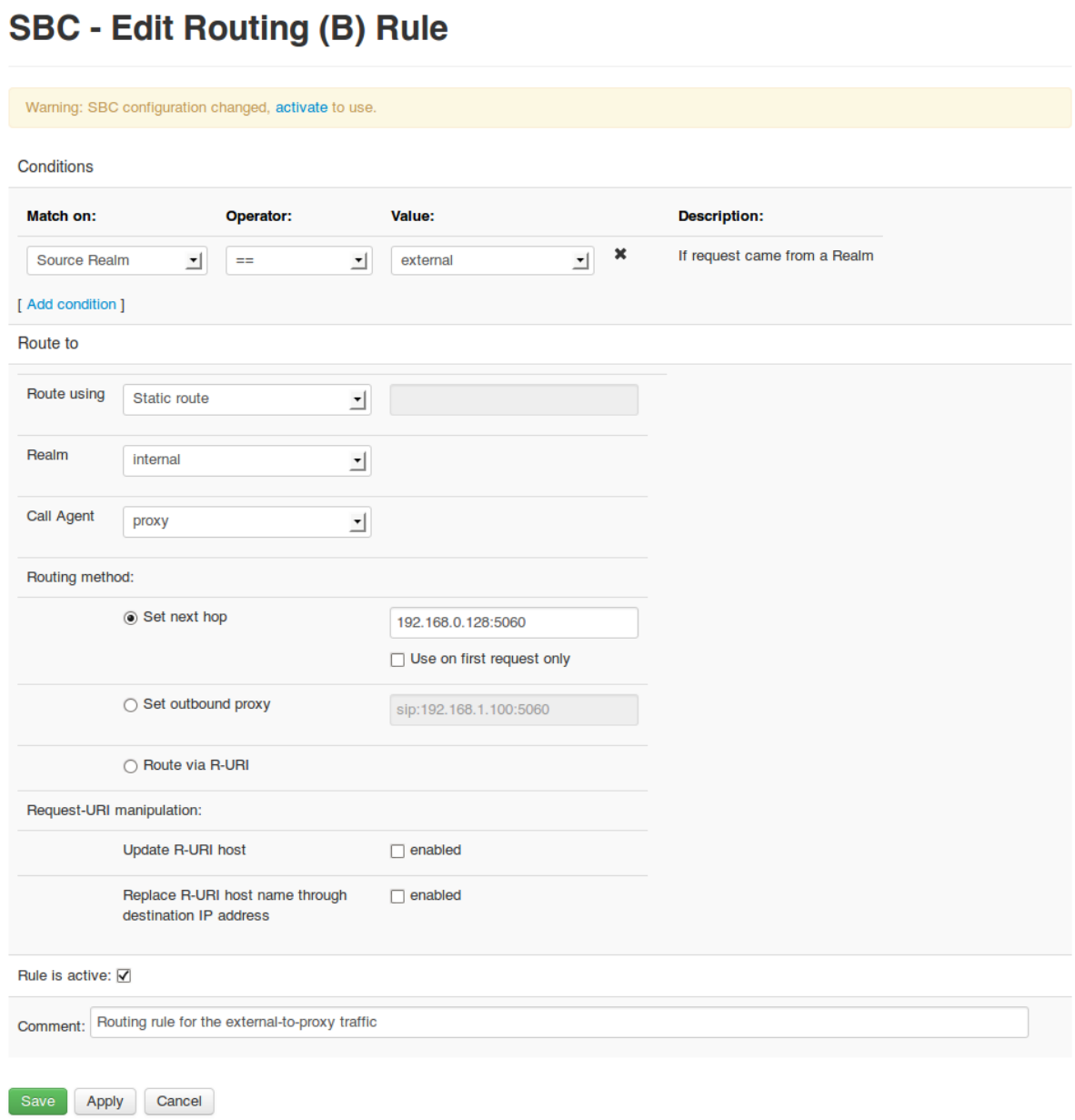


Fig. 20: Routing Rule for external to internal traffic

Nevertheless, if needed it can use more sophisticated matching criteria to specify routing decisions: It can divert Message-Waiting-Indication traffic to a different server based on SIP method, different media servers based upon codecs in use, different destinations based on custom-defined header fields, and so on, and so forth.

6.7 SIP Mediation

SIP Mediation features of the ABC SBC allow administrators to introduce massive changes to the signaling protocol. This is often necessitated by devices with imperfect SIP support, differing practices such as dialing plans between peering providers, or need to implement network-based services such as Private Asserted Identity (RFC 3325).

The actual mediation rules are placed in inbound and outbound rules. The inbound rules are used to modify incoming traffic coming from a Realm or a Call Agent to comply to local policies. For example, the inbound rules may transform telephone numbers from a local PBX's dialling plan to the global E.164 standard. All subsequent actions already work with modified SIP messages. The outbound rules are used to modify outgoing traffic to

a form that the receiving Call Agent can or shall process. For example the outbound rules can remove all but low-bandwidth codecs for the target known to be on a low-speed link.

It needs to be understood that mediation is a double-edged sword: massive changes to the signaling protocol can, if not configured properly, cause substantial harm to interoperability. If the ABC SBC encounters, that a SIP message modified by mediation rules breaks standard too far (such as if it generates an empty header-field), it discontinues processing of the message and sends a 500 response back. Still many changes may be syntactically legitimate, remain undetected and result in impaired interoperability.

This section discusses mediation of the signaling protocol, SIP. Mediation of media, that includes codec negotiation and transcoding, is documented in the section *Media Handling*.

6.7.1 Why is SIP Mediation Needed?

There are multiple root causes why SIP devices have often troubles communicating with each other. There are different standardization groups working on SIP. Different developers often interpret the same specifications differently. Operators deploy different operational and naming practices.

The ABC SBC has the capability to overcome some of these interoperability problems by manipulating the content of SIP messages so that they better fit the expectations of the receiving side. One can distinguish between several frequent interoperability issues: compatibility between various SIP protocol extensions, dealing with deviations from the specification and best current practices caused by non-compliant devices and operating procedures, and incompatibility between different transport protocols used for conveying SIP signaling.

SIP Standard Extensions:

There are various flavors of the SIP protocol. Even the basic SIP IETF standard is extended by tens of accompanying specifications, some of them are deployed, some of them not. Several other standardization bodies have chosen to add even more extensions specific to their use of SIP. In the fixed environment, the *TISPAN specifications <http://www.etsi.org/tispan/>* are used. In the mobile network environment the *3GPP IMS specifications <http://www.3gpp.org/>* are the most favoured. *SIP-I <http://www.itu.int/rec/T-REC-Q.1912.5-200403-I/en>* is proposed for trunking scenarios in which SIP is used as the signalling protocol used to connect SS7 based networks over an IP core network.

> The differences between the SIP specifications from IMS, IETF and TISPAN are mainly restricted to the addition of certain headers, authentication mechanisms and usage of certain SIP extensions such as NOTIFY/SUBSCRIBE or certain XML body formats.

In the context of interoperability of SIP flavours, the ABC SBC can provide the following services:

- Stateless SIP header manipulation: The ABC SBC can be configured to remove certain headers and add others. This way, The ABC SBC can for example delete headers that are useful in an IMS or TISPAN but not in an IETF SIP environment.
- Message blocking: Certain SIP messages might be useful in one network as they provide a certain service. However, if this service is not provided across the interconnection points then exchanging them across the networks does not make sense. SBCs can be configured to reject certain messages such as NOTIFY if presence services are not provided across the network for example.

Deviations from the SIP Standard and Best Practices:

The experience from various interoperability events shows that different vendors interpret the SIP specifications slightly differently. Especially parts that are specified with the strength of “SHOULD” or “MAY” are often implemented as a “MUST” or ignored completely. This makes the communication between two components from different vendors sometimes impossible. Sometimes even if the SIP equipment implements the standard correctly, operators practices for deploying SIP differ to the extent that the protocol needs to be fixed.

The ABC SBC can be configured to overcome some of these issues and to fix certain issues that cause these interoperability problems by offering the following features:

- Existence of certain headers: Some SIP components expect to see certain SIP headers with certain information, for example a *Route* header pointing to them. Others might not bother to add this header. The ABC SBC can be configured to take these special interpretations of the implementers into account before forwarding a request and add or remove problematic headers.

- Location of information: Some SIP components expect to see their address in the Request-URI whereas others want to see it in the *Route* header or both. This might not always be how the location information is included in the SIP request especially if a request was redirected from one component to another.
- Tags and additional information: Again some SIP components might expect to see certain tags and parameters attached to certain headers such as **rport** with a *Via header* whereas other SIP components might not add them.

SIP Transport:

SIP can be transported over UDP, TCP and TLS. The capabilities of different SIP implementations might vary with this regard. That is, some components could support UDP but not TCP and others prefer to use TLS. Therefore, the ABC SBC can be used to convert the transport protocol used by the source to the transport protocol preferred by the destination.

6.7.2 Request-URI Modifications

The most common manipulation is that of request-URI. Request URI describes who should receive the SIP request. It may include an E.164 telephone number (like sip:+1-404-1234-567@pbx.com), a PBX number (sip:8567@pbx.com) or be formed as an email-like address (sip:amadeus@mozart.at). A typical reason for changing the request URI is normalisation of different dialling plans. As an example you may translate a local extension number (768) for a PBX with prefix (+1-404-1234) into a globally routable E.164-based URI sip:+1-404-1234-567@national-gateways.com. You can use several types of modifications to the request-URI, all of them are applied only to the first session's request. The most important request-URI actions are the following:

- **Strip RURI user:** strips the specified number of leading characters from the user part of request URI. For example strip-RURI-user(1) applied to the PBX URI 8567@pbx.com yields the extension sip:567@pbx.com without the local "8" prefix. The action is applied as many times as it is called.
- **Prefix RURI user:** inserts a prefix to the user part of request URI. For example, prefix-URI("+1-404-1234-") applied to the URI from the previous step yields sip:+1-404-1234-567@pbx.com. The result is accumulated if the action is applied several times.
- **Append to RURI user:** appends a suffix to the user part of request URI. The parameter takes suffix value. It may include replacement expressions. The result is accumulated if the action is applied several times.
- **set RURI:** entirely replaces the request URI with a new value.
- **set Contact URI host:** entirely replaces the Contact URI host with a new value. Note that the update isn't run on REGISTER replies.

Also note that the resulting URI not only describes the recipient, but its host part is used to determine the next hop IP address if a route is used with the **Route via R-URI** option.

It is also worthwhile mentioning that URIs often represent additional services a caller gets. For example if a caller prefixes number of an O2 subscriber in Germany with 33, his call will be directly routed to the recipient's voicemail. However administrators would be ill advised to overload request URI with more than routing functionality. An infamous example is using a plain-text password as phone number prefix for authentication. The fraudster *Edwin Pena <http://www.fbi.gov/newark/press-releases/2010/nk020310a.htm>* found that out, yielded more than 10 million minutes of VoIP service and in 2009 eventually two years in federal prison.

Several other mediation actions can process sub-parts of request-URI. They include:

- **Set RURI host**
  - Replace host(:port) part of Request-URI with a new value specified in the GUI.
  - Parameters: new host or host:port
- **Set RURI parameter**
  - Add or replace parameter of Request-URI.
  - Parameters: RURI parameter name, RURI parameter value
- **Set RURI user**

  - Replace user part of Request-URI with a new value.
  - Parameters: new user part.
- **Set RURI user parameter**
  - Add or replace parameter of user part of Request-URI.
  - Parameters: parameter name, parameter value.

6.7.3 Changing Identity

Identity of SIP session participants is also described in many other SIP header fields that sometimes need to be changed.

Every SIP request must include URIs of session initiator in the *From* header-field and URI of intended recipient in the *To* header field. The SIP standard has intended to use the *From* and *To* header field only as informational description of how a session was started . URI of the originator in the *From* header field has limited identity value as the plain-text URI is not covered by a message integrity check and can be easily changed by elements in the SIP-path. Even a user client is quite free to put anything in the URI unless there is a client's outbound SIP proxy enforcing specific address for a digest-verified caller.

The URI in *To* header-field may have little relation to the actual recipient of a SIP request as the actual next hop is stored in the request URI.

Notwithstanding how "light-weight" information *From* and *To* header fields convey, some operators deploy policies based on them. They may only accept requests with *From* and *To* URIs that comply to their local convention. There were even cases when *To* URI was used for routing. Therefore it is often useful to modify *To* and *From* header fields. These modification rules apply to the first request of a SIP dialog. *From* and *To* in all subsequent messages of a session are transformed transparently in compliance with the SIP protocol specification. The most important *To* and *From* changing actions are the following:

- **Set From / Set To** :replaces the whole *From*/*To* header field with a new value, for example "Jasmine Blue" sip:jasmine@blue.com. Only "tags" in the *From*/*To* header-fields remain unchanged to guarantee unique identification of SIP dialogs.
- **Set From User / Set To User**: replaces the user part of the From/To URI.
- **Set From Host / Set To host**: replaces the host(:port) part of URI with a new value
- **Set From / To display name**
  - Set only the display name of the *From* / *To* header.
  - Parameter: new display name.

Additionally, the SIP protocol is using digest authentication identity (**RFC 2617**) to verify who is initiating a request. If the digest identity of a request originator needs to be changed, the action **UAC auth** is used. It takes the following parameters needed for the authentication procedure: username, password and realm. A request forwarded downstream and challenged to authenticate by a downstream server is then resubmitted by the ABC SBC using these credentials.

Substitution Expressions

SIP message modifications typically "glue" pieces of the original messages and intended changes. For example, a new *To* URI is to be formed using destination's hostname (say "target-gw.com") and telephone number in request URI (say "+1-404-1234-567"). The corresponding **set-To** action needs to access the telephone number in the original request. To address cases like this, the mediation parameters may refer to elements of the original message by so called *Replacement expressions*. These always begin with a **$** character. In our example, the user part of the request URI is referred to as "**$rU**" and the action has the form:

> Set To ("sip:\protect\T1\textdollarrU@targetgw.com").

Other important replacement expressions are **$fu** for *From* URI, **$tu** for *To* URI, **$si** for source IP address, **$H(headername)** for value of a header field.

If you need to access some sub-parts of the original SIP message without an addressable name, simple substitution expression are not enough. Then regular expressions have to be used to select them. This is called „*regexp backreferences*. The backreference expressions refer to parts of SIP messages that were matched in rules' conditions. For example, to access the protocol discriminator in a URI, you need to create a rule condition matching it using regular expression, and then refer to the matched expression. You would be forming a rule like this:

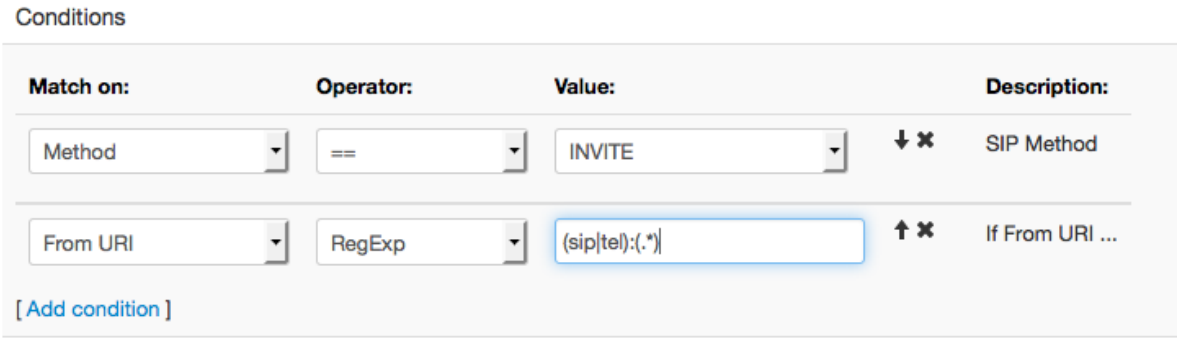


Fig. 21: Example of a Condition Being Referred to by a Backreference Expression

the second condition's first sub-part (i.e. matched by the expression in the first parentheses) of the regular expression would identify the protocol discriminator and yield "sip" for SIP URIs. The expression would be formed as this

```
$B(2.1)
```

6.7.4 SIP Header Processing

URI adaptation shown in previous paragraphs is important for harmonization or routing and identity representation between different SIP devices and administrative domains. Yet there are many other header fields conveying important information in need of adaptation. Worse than that, some of them are not even known at the time of writing this documentation. That's because some of them may be proprietary – for example Sipura SIP phones add QoS reports to every BYE message they send. Some header fields may even be specified in recently published standard. Yet even then the ABC SBC can help – it can use general purpose text-processing methods thanks to SIP's text-based nature. Particularly the following actions are available:

- **Remove Header:** Remove-header removes all occurrences of a header-field identified by its case-insensitive name from all requests and responses in a session. Exceptions apply: mandatory header-fields are not removed: CallID, From, To, CSeq, Via, Route, Record-Route and Contact. If a header-field with compact name form occurs, both forms must be removed explicitly. Newly added header-fields are not removed by this action.
- **Set Header Blacklist:** is a convenience function removing multiple header fields by a single action. It takes comma-separated list of header-field names as parameter and achieves the same effect as if you used multiple occurrences of the Remove-Header action. Blacklists are applied one by one in the order in which they appeared in the rules and are executed after applying both A and C rules. Blacklists can be a nice short-cut for removing a header-field which has both normal and compact name. For example, you may want to configure deletion of both forms of the Subject header field by using

  ```
  Set-Header-Blacklist("Subject,s")
  ```

- **Set Header Whitelist:** is an even more aggressive convenience function for removing multiple header fields. If used, all but mandatory and whitelisted header fields are removed from all requests and responses belonging to a session. Tje action is applied after processing of both A and C rules completes.

- **Add Header:** adds a new arbitrary header-field to a dialog-initiating request. This action only applies to the first request of a session. Its greatest power comes from the ability to craft complex header-fields using the substitution expressions.

SIP Header Modification Examples

Let us show the power of these actions on an example. A real-world case is translation of identity between the pre-standard *Remote-Party-ID* header field, still used by some SIP equipment, and the standardized *Asserted Identity*, see **RFC 3325**. Both fulfill the same purpose, yet differ in their syntax which needs to be translated from one form into the other.

The pre-standard header-field looks like this

```
Remote-Party-Id: "Mr. X" <sip:+1-404-1234-000@sipsip.com>;privacy=full
```

The standard form looks like this

```
P-Asserted Identity: "Mr. X" <sip:+1-404-1234-000@sipsip.com>
```

The simplest way for translation is

- finding out if there is an occurrence of *Remote-Party-Id* header-field by a rule with condition

  ```
  if Header(Remote-Party-Id) does not match RE ^$"
  ```

- removing the header field by action

  ```
  Remove-Header(Remove-Party-Id)
  ```

- and eventually forming the newly crafted header field using the URI in the previous header-field by

  ```
  Add-Header(P-asserted-identity: $Hu(remote-party-id)"
  ```

Note that while this example mostly works, it ignores some parameter details for sake of brevity.

It is important to keep in mind that mediation changes have impact on subsequent SIP processing: replacement expressions and header-field tests in condition consider the changed value.

The following example rules change request URI and From URI.

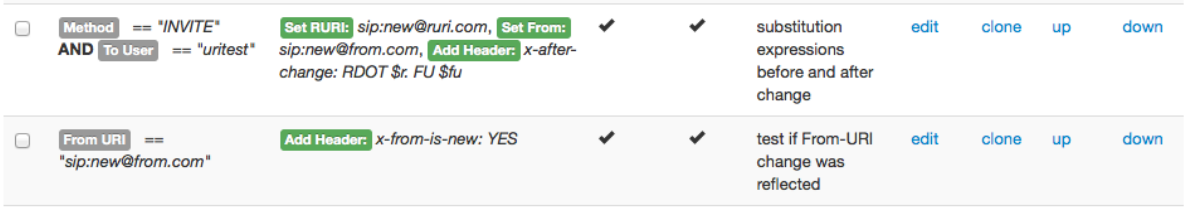


Fig. 22: Impact of Mediation Changes on Subsequenet Processing

Substitution expressions in the *Add Header* Action print the request URI and From URI in a troubleshooting header-field named *x-after-change. Because they refer to the **current** value, the new URIs appear in the outgoing INVITE regardless what they included originally. Similarly, the header-field value condition that tests if the From URI has assumed the new value prints YES in another troubleshooting header-field name *x-from-is-new*:

```
INVITE sip:new@ruri.com SIP/2.0.
Via: SIP/2.0/UDP 192.168.0.84;branch=z9hG4bKtwvtoaKk;rport.
From: <sip:new@from.com>;tag=170A7805-533943A10009B434-D85F9700.
To: <sip:uritest@abcsbc.com>.
CSeq: 10 INVITE.
```

(continues on next page)

(continued from previous page)

```
Call-ID: 77443978-533943A10009B47B-D85F9700.
User-Agent: Blink Lite 3.1.1 (MacOSX).
x-after-change: RDOT sip:new@ruri.com FU new@from.com.
x-from-is-new: YES.
....
```

Option tags

Option tags are unique identifiers used to designate extensions in SIP. These tags are used in Require and Supported header fields.

To simplify manipulation with these headers ABC SBC offers since version 4.5 following conditions:

- **Supported header**: allows to check whether an extension is (is not) present in Supported header field.
- **Require header**: allows to check whether an extension is (is not) present in Require header field.

and actions:

- **Update Supported header**: allows to add option tags (*Add tags*) or remove them (*Remove tags*) from Supported header field or overwrite them completely (*Set tags*)
- **Update Require header**: allows to add option tags (*Add tags*) or remove them (*Remove tags*) from Require header field or overwrite them completely (*Set tags*)

6.7.5 Early Media, Ring Back Tone and Forking

In SIP, so called "early media" and "forking" are quite complex SIP features which make interoperability sometimes a challenge, especially when occurring together.

Early media appeared in the SIP protocol as a PSTN backwards-compatibility feature. In PSTN the difference between early media like "please wait your call is important to us" and the actual call is simple: the latter is charged for, the former is not. This is by the way the reason, why "early media" is sometimes humorously referred to as "late charging". Early media appear often when the called party is a PSTN gateway. The same protocol vehicle is often also used to implement "ring back tone". The protocol flow is rather simple: The callee sends a provisional response with a reply code equal to *180* or *183* including an SDP answer and starts sending RTP with the ring back tone to the caller. Usually, the caller User Agent only starts rendering the ring back tone to the user when this response is received. The protocol usage examples and details are well described in the **RFC 3960**.

Forking is a feature anchored in the SIP specification **RFC 3261**. It permits SIP proxy servers to forward one incoming requests to multiple different destinations. For example, one can setup this way all his phones to ring in parallel: soft-phone, hard-phone, smart-phone and even a PSTN phone behind a PSTN gateway. Forking can occur in parallel or in series. If serial forking is used, a forking proxy following best current practices sends a 181 inbetween. The "forked" INVITE requests may look almost identical but each of them always must have a unique "branch" identifier in the topmost Via header field.

Various unpredictable situations appear when forking and early media appears at the same time. For example two PSTN gateways send both early media to the caller. To deal with such situations the ABC SBC does only accept the first early media stream and discards the subsequently received ones.

The actions described in this Section help to customize the behavior of the ABC SBC to some special cases.

The action **Drop SDP from 1xx replies** drops SDP payload from all listed 1xx SIP answers. The action takes as parameter a comma-separated list of reply codes. SDP payloads are dropped from all responses with any of these codes. This action is especially useful if specific replies should be handled, for example a locally generated ring back tone should be preferred to a ring back tone from the far end. Note that the RTP relay is not started if all provisional response are dropped, i.e. a provisional response needs to be processed for the RTP relay to be initialized, also for relaying early media.

Fig. 23: Drop SDP from reply

Another action **Drop early media** drops the RTP packets of early media, that is until the call is established. Note that if early media shall be dropped from signaling entirely, the actions "Drop SDP from 1xx replies" in combination with "Translate reply code" 183->180 must be used.

Fig. 24: Drop early media

Support serial-forking proxy: This action allows to reset early media when a downstream SIP proxy server indicates by a 181 response that it has chosen to try some other destination for the call. By default, only the first early media arriving to the SBC is permitted, all other early media is dropped. This strict policy assures that downstream SIP forking cannot create multiple early media streams mutually interfering with each other. With this option, one can make an exception to the rule and permit early media coming later to override the previously established early dialog. It works safely as long as there is no parallel early media and 181 indicates that a later early media stream legitimately replaces the previous stream.

The following SIP flow-chart from Section 2.9 of **RFC 5359** show a situation in which a SIP proxy generates a 181

```
Alice            Proxy         User B1             User B2
  |                |              |                   |
  |    INVITE F1   |              |                   |
  |--------------->|   INVITE F2  |                   |
  |(100 Trying) F3 |------------->|                   |
  |<---------------|180 Ringing F4|                   |
  | 180 Ringing F5 |<-------------|                   |
  |<---------------|              |                   |
  |                 Request Timeout                   |
  |                |              |                   |
  |                |   CANCEL F6  |                   |
  |                |------------->|                   |
  |                |   200 OK F7  |                   |
  |                |<-------------|                   |
  |                |     487 F8   |                   |
  |                |<-------------|                   |
  |                |     ACK F9   |                   |
  |                |------------->|                   |
  |(181 Call is Being Forwarded) F10                  |
  |<---------------|              |    INVITE F11     |
  |                |--------------------------------->|
  |                |              | 180 Ringing F12   |
  | 180 Ringing F13|<---------------------------------|
  |<---------------|              |      200 OK F14   |
  |                |<---------------------------------|
  |   200 OK F15   |              |                   |
  |<---------------|              |                   |
  |     ACK F16    |              |                   |
  |--------------->|              |     ACK F17       |
  |                |--------------------------------->|
  |               Both way RTP Established            |
  |<================================================>|
  |    BYE F18     |              |                   |
  |--------------->|              |      BYE F19      |
  |                |--------------------------------->|
  |                |              |    200 OK F20     |
```

(continues on next page)

(continued from previous page)

```
|  200 OK F21     |<---------------------------------|
|<----------------|                  |               |
|                 |                  |               |
```

The action **Fork** allows to add a new branch to a processed request and start forking. Multiple occurrences of the action result in multiple branches of the request. The action takes only one parameter, the request URI of the forked request. The parameter can use replacement expressions, however if an invalid SIP URI is formed the call will fail.

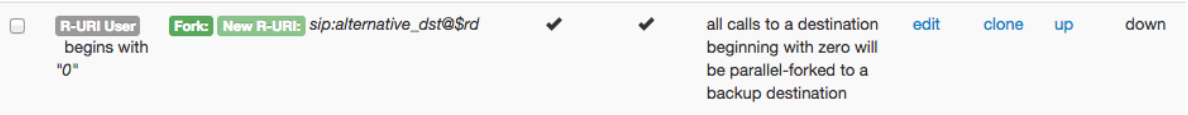


Fig. 25: Forking

6.7.6 Call transfers

Using the action **Call transfer handling** it can be configured how in-dialog REFER requests are handled in the ABC SBC. The configuration is per call leg, i.e. if used in inbound (A) rules REFER handling is set for the A leg, if used in outbound (C) rules it is set for the B leg.

Following methods of REFER handling can be used:

- **pass-through**

  Pass REFER through the ABC SBC to the remote peer (default).

- **reject**

  Reject the REFER request with a `403 Forbidden` reply.

- **handle internally**

  In case of an attended call transfer to another call established through the ABC SBC (REFER with `Replaces` in `Refer-To` pointing to a local call) the call legs are connected locally. Only offer-answer exchanges (re-INVITEs) that synchronize session description on both ends are generated.

  In case of an unattended call transfer (no `Replaces` in `Refer-To`) the ABC SBC generates a new INVITE to the requested destination. This INVITE can be handled in routing (B) rules and outbound (C) rules similarly to regular calls. For detection of such locally generated calls the condition **Request source** can be used.

  In case of an attended call transfer to a non-local call (`Replaces` in `Refer-To` refers to a non-existent call leg) the ABC SBC generates a new INVITE with `Replaces` to the requested destination. This INVITE can be handled the same way as an INVITE generated for an unattended call transfer mentioned above.

Limitations:

- only in-dialog REFER requests are handled
- attended call transfer is not possible with transparent call IDs

6.7.7 INVITE with Replaces handling

ABC SBC is able to handle INVITE with Replaces header locally, if the Replaces header points to a call established on the SBC.

The action **Handle INVITE with Replaces header** is used for this purpose - it activates local INVITE with Replaces handling.

Limitations:

- INVITE with Replaces can not be handled when replacing call with transparent call IDs

6.7.8 Mapping Dialog-IDs in INVITEs with Replaces

If an INVITE with Replaces passes the ABC SBC, and the call to be replaced is also traversing the SBC, with transparent call IDs not enabled the Dialog Identifiers in the Replaces header refer to the call leg on the side before the SBC, but do not have a meaning after the SBC.

Using the *Map Replaces header* action, the dialog identifiers are replaced with the corresponding ones on the other side of the SBC so that the Replaces still is valid.

6.7.9 Other mediation actions

The ABC SBC supports various actions related to SIP processing:

- **Enable transparent dialog IDs**
  - Use the same dialog identifiers (Call-ID, From-tag, To-tag) on both sides of a call (e.g., for the incoming and out going messages). If this action is not enabled, the FRAFOS ABC SBC changes dialog identifiers. Unchanged Call-ID may be a security concern because it may contain the caller's IP address. However, transparent identifiers make troubleshooting and correlation of call legs much easier. Also, for call transfers using REFER with *Replaces* as used in call transfer scenarios to work through the SBC, transparent dialog IDs need to be enabled.
- **Forward Via-HFs**
  - This option makes the SBC keep all *Via* headers while forwarding the request. This behavior mimics what a proxy would do, especially in combination with the **Enable transparent dialog IDs** action (the only remaining difference to a proxy is the non-transparent *Contact* header field). Note that forwarding the *Via* headers exposes the IP addresses of entities on the incoming leg side of the request.
- **Translate reply code**
  - Change SIP response code and reason for all SIP responses with a specific code. Note that changing responses between SIP reply classes may seriously break proper operation.
  - Parameters: SIP response code to change, SIP code and reason phrase to use for the reply sent out.
- **Allow unsolicited NOTIFYs**
  - The ABC SBC keeps track of subscriptions and usually only lets NOTIFY messages through if a subscription for it has been created before (through a SUBSCRIBE or a REFER). This action tells the SBC to let pass NOTIFY messages even if no subscription has been created before.
- **Relay DTMF as AVT RTP packets (RFC4733/RFC2833)**
  - relays DTMF tones as RTP avt-tones packets (**RFC 4733**/**RFC 2833**)
  - Parameters: none
- **Relay DTMF as SIP INFO**
  - relays DTMF tones as proprietary SIP INFO payload
  - Parameters: none

- **Diversion to History-Info**
  - converts SIP diversion header-field (RFC 5806) into the History-Info header-field (RFC 4244) using the guidelines set in RFC 6044.
  - Parameters: none
- **Set Max Forwards**
  - sets the value of Max-Forwards header field in forwarded SIP requests to the configured value. This limits the number of hops a request can be forwarded until it is bounced back. It may make sense to set it to a lower value than RFC 3261 recommends (70). If this action is not used, the value in incoming request is decremented by one before forwarding.
  - Parameters: number of hops.
- **Set Content Type whitelists** and **Set Content Type blacklists**
  - limits SIP content types to well known payload types (whitelists) or to all but specifically prohibited payload types (blacklists). Most VoIP SIP requests include the type of *application/sdp*.
  - Parameters: comma-separated list of content-types
- **Add dialog contact parameter**
  - Allows to add a parameter to the contact URI generated by the SBC.
  - Parameters: The side of the call (caller/A leg, callee/B leg) can be specified, the parameter name and value.
- **Set Contact-HF parameter whitelists** and **Set Contact-HF parameter blacklists**
  - defines which Contact HF parameters are forwarded through the ABC SBC . By default no parameters are forwarded. With whitelisting, only specified parameters are forwarded. With blacklisting, all but specified parameters are forwarded.
  - Parameters: comma-separated list of Contact parameter names
- **Forward Contact-HF parameters**
  - makes sure all Contact HF parameters are forwarded as received in incoming request. If no action is used, no parameters are forwarded at all.
  - Parameters: none
- **Call transfer handling**
  - The actions defines in which mode incoming REFERs will be processed. They are either rejected, forwarded or handled locally.
  - Parameters: REFER-processing mode

6.8 SDP Mediation

SDP mediation allows to manipulate how applications codecs will be selected during session negotiation.

6.8.1 Codec Signalling

In SIP call parties are free to negotiate their capabilities using the offer-answer model, see **RFC 3264**: The caller offers its capabilities such as supported codecs and the caller party matches those against its own. In some cases it may be reasonable to restrict the list of offered codecs. Mostly, this is done when there are bandwidth constraints.

If media anchoring is used, every single media stream enters and leaves the SBC. With the most common but "hungry" codec G.711, it means 172 kbps x 2 in each direction, which corresponds to a maximum of about five thousand calls on a gigabit link (the actual limit is in fact even lower due to packet rate constraints).

G.729 is probably the most widely used codec with lower bandwidth consumption. The bit rate for G.729 yields 62 kbps in each direction (the rate includes UDP, IP and Ethernet overhead).

For mobile clients, bandwidth hungry codecs with large packet size like G.711 can pose additional problems: Due to longer use of the wireless interface, battery life is reduced, and also the packet loss rate is greatly increased with the bigger packet sizes. On the other hand, CPU intensive codecs may also strain the battery on mobile clients if they are not implemented in hardware.

For these reasons, the ABC SBC offers you these functions

- setting codec preferences (**Set codec preference** action). Specifies in descending order which codecs offered in SDP payload should be "picked".
- transcoding (**Enable transcoding** action) – allows to convert sender's media from encoding the received does not support to encoding he does. See more in Section *Transcoding*.
- codec white/blacklisting (**Set codec whitelist** and **Set codec blacklist** actions) explicitly specifies which codecs are permitted or not.

In order to save bandwidth and improve battery life and call quality, to mobile clients G.711 should not be used by using a **Set codec blacklist** action with "PCMU,PCMA" as blacklisted codecs.

Another example is emergency calls (911), where due to call quality concerns G.711 is the mandatory codec. If, for bandwidth saving reasons, the G.711 codec is usually blacklisted, it should be whitelisted for calls sent to an emergency gateway.

If codec restrictions result in a failure to find a common codec, the ABC SBC offers you to use built-in software based transcoding to increase interoperability.

Please refer to the Media processing section, see Sec. *Media Handling* for a complete reference of functionality the ABC SBC offers to restrict the set of used codecs, give certain codecs a preference or transcode between codecs.

6.8.2 Media Type Filtering

For an audio call, the media type in the SDP is "audio". For a normal video call with audio, the two media types "audio" and "video" are negotiated, for other types of calls ("image", screen sharing etc), other media types are possible.

Media types may be filtered using the actions

- **Set media blacklist** - remove all blacklisted media types from SDP
- **Set media whitelist** - remove all but whitelisted media types from SDP

The media blacklist/whitelist actions have as parameters a comma-separated list of media types (audio, video, image, ...) to be blacklisted. It is applied to all SDP messages exchanged at any time during the call. In the case that after applying the action no media type is left in the SDP message then the request will be rejected with a response message *488*.

Example: Allow only audio payload to pass and prevent video streams to be negotiated; for the User Agents it will appear as if the other side does only support audio.

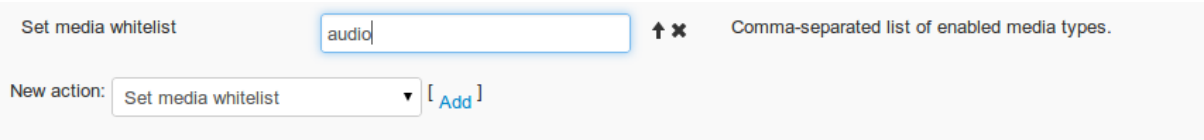


Fig. 26: Remove video streams

Example: Let audio and audio/video calls through.

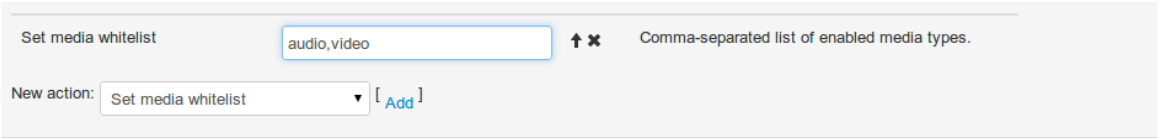


Fig. 27: Allow only audio and video

Example: Remove “image” media type.

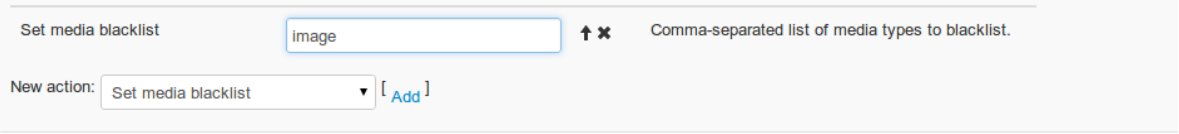


Fig. 28: Do not allow exchange of images

6.8.3 CODEC Filtering

The actual audio or video content of a call can be encoded with different codecs, which have each different properties regarding:

- audio or picture quality
- bandwidth consumed
- latency introduced
- processing power required
- resilience regarding packet loss

For example, the G.711 codec has “toll quality” (audio quality roughly equivalent to PSTN, 8khz sampling rate/narrowband) at 64kbit/s (roughly 80 kbit/s including headers in each direction), introduces low latency, requires little processing but is not resilient against quality degradation with packet loss. The G.729 codec has a bit less than “toll quality” at 8kbit/s, 6.4kbit/s or 11.8 kbit/s (depending on the used annexes) with modest latency introduced, some processing power required, and some resilience against packet loss.

In order for two endpoints to successfully establish a call, both endpoints need to support the same codecs. The codecs actually used in a call are negotiated using the SDP protocol and the SDP offer/answer method to the subset of codecs supported by both endpoints, and thus it is usually best to let the endpoints negotiate with the most options possible.

If for some reasons codecs need to be filtered, the actions

- **Set CODEC whitelist** - remove all but whitelisted media types from SDP
- **Set CODEC blacklist** - remove all blacklisted media types from SDP

are used. Each of these actions takes a comma-separated list of codecs to white- or blacklist, which must be the names of the CODECs as they are used in SDP[1] . Codec names are case-insensitive, a blacklist of "g729,ilbc" is equivalent to "G729,ILBC". In the case that after applying the action no codecs are left in the SDP message then the request will be rejected with a response message *488*.

Fig. 29: Setting a whitelist

Fig. 30: Setting a blacklist

All of the white- and blacklists applied for a call are executed one after another. For example, if one action sets the whitelist "PCMU,PCMA,SPEEX" and another action sets the whitelist "PCMA,G729", it would result in only PCMA (G.711 alaw) be let through for the call.

The codec white and blacklists are applied on both legs, the incoming and the outgoing call legs, and on all SDP messages going in both directions (from caller to callee and from callee to caller) at any time during the call.

6.8.4 CODEC Preference

Codec negotiation in a SIP call usually works this way

- the offerer (the caller in calls with normal SDP offer-answer negotiation; the callee in calls with delayed SDP offer-answer negotiation) lists the supported codecs for a call in preferred order, so the first codec in the SDP is the codec preferred by the offerer
- the other party (the answerer) selects from this list the subset of codecs that it supports, and orders it according to its preference or local policy, it may for example accept the order that the offerer asked for
- both parties encode and send media with the codecs that are in this subset, and usually the first codec in the answer is used, but (even without re-negotiation) any party may switch in-call to any codec in this subset

Because the User Agents usually respect the codec preference of the other side, the operator may influence the codec actually used for a call in the SBC by reordering the codecs as they are listed in the SDPs. Codec preferences may be influenced in order to

- improve audio quality by preferring better performing codecs
- save bandwidth
- save processing power on the User Agents, e.g. especially if mobile/battery powered devices are used

The **Set CODEC preferences** action has as parameters two comma-separated lists of codecs, for the A (caller) leg and the B (callee) leg respectively. An entry may have the specific sample rate appended separated with a slash, e.g. speex/32000, speex/16000, speex/8000

[1] e.g. PCMU for G.711 ulaw, PCMA for G.711 alaw. See http://www.iana.org/assignments/rtp-parameters and the IETF payload type specifications (RFCs) for the used names of the CODECs.

Fig. 31: Setting the codec preferences

Any codecs in this list found in the SDP messages exchanged in either direction are prioritised in the order listed, by placing them at the beginning of the codec list in the SDP document. It is a good practice to configure the same order for both legs.

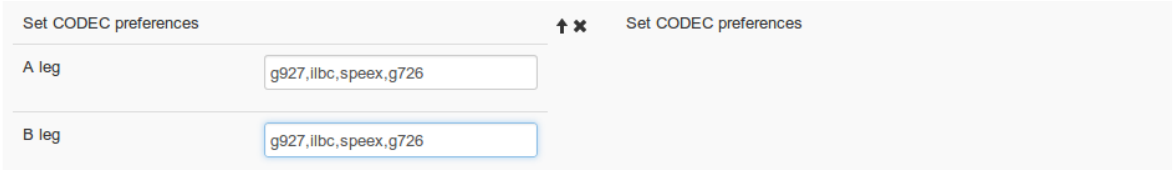


Fig. 32: Example: Prefer bandwidth-saving codecs (codecs that compress more): G.729,iLBC,speex,G.726

Fig. 33: Example: Prefer codecs with high audio quality: OPUS,SILK,speex/32000,speex/16000,G.722,AMR-WB

Additionally codec attributes offered in SDP can be filtered out using actions **Set SDP attribute whitelist** and **Set SDP attribute blacklist**.

6.8.5 SDP Bandwidth attribute limiting

The SDP may contain a (session-level, or media-level) b=<modifier>:<value> attribute, which sets the bandwidth to be used (see RFC4566). Different types of bandwidth signaling are standardized, denoted by different modifiers; the most common being TIAS (RFC3890), AS (application specific, RFC4566) and CT (conference total, RFC4566). The action **Set SDP bandwidth limit** can be used to limit the signaled bandwidth: If there is a bandwidth attribute for the specified type, it will be set to the limit if it is signaled to be more than the limit. If there is no bandwidth attribute for the specified type, one will be added.

Especially when the actual RTP bandwidth available to a call is limited using the action **Limit Bandwidth per call (kbps)**, using this action the SBC can signal the maximum available bandwidth to the endpoints.

If 'Media type' is not set, this action sets the session-level bandwidth attribute. If 'Media type' is set, it sets the media-level bandwidth attribute for that media type. E.g., if 'video' is set as Media type, then all m-lines with type 'video' will have a properly limited bandwidth attribute. 'Media type' can be set to only a single media type value (i.e. 'video' or 'audio'), no list can be given here (i.e. 'video, audio' is wrong); if multiple media types should be limited, multiple actions must be used.

6.9 Media Handling

6.9.1 Introduction

In SIP networks, the signalling and media packets may traverse different paths and may be handled by different servers in the path. The ABC SBC can be on both the signalling path and the media path, or only on the signalling path of a call.

In the SIP signalling, the IP addresses between which the actual media is exchanged is negotiated using Session Description Protocol (SDP). The default signaling mode establishes a direct media path between the two call parties as shown in Chart I of Figure *RTP Anchoring with and without Symmetric Mode*. If the ABC SBC is configured to intervene and insert itself in the media path, it replaces IP addresses in SDP signaling with its own, attracts RTP packets to itself and forwards them to the other call party, as shown in Chart I and II.

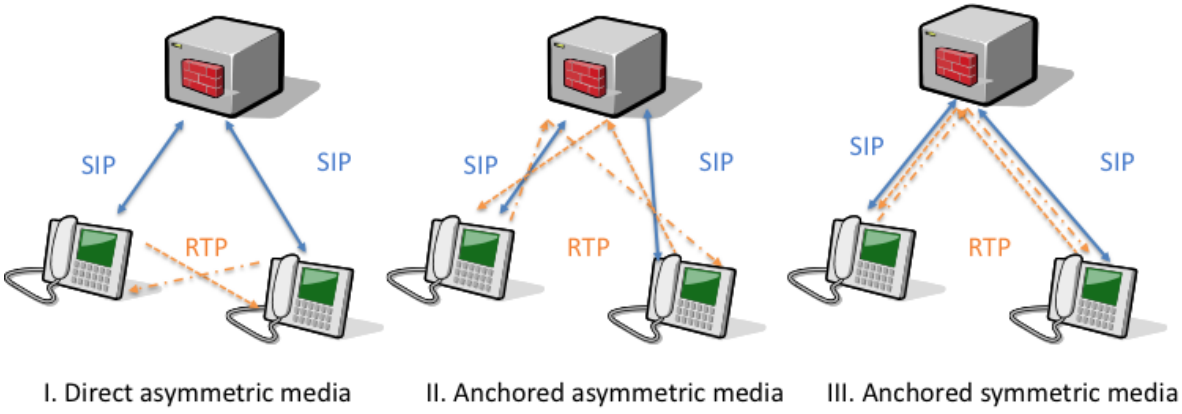


Fig. 34: RTP Anchoring with and without Symmetric Mode

Generally it is desirable to have RTP processed and relayed at as few network elements as possible, in order to maintain lowest possible total latency and delay variations (jitter). Performance impact of media relay is discussed in more detail in Section *SBC Dimensioning and Performance Tuning*. However, the ABC SBC must be inserted in the media path in the following quite common situations:

- **NAT handling** - User Agents that are behind a NAT are not able to send RTP directly to other User Agents behind NAT
- **Connecting unroutable networks** – when the ABC SBC connects networks that cannot directly send packets between themselves, media anchoring must be enabled.
- **Topology hiding** - to improve end point and network security, the ABC SBC prevents entities from learning the addresses of other entities in the network
- **Bandwidth limitations** - the ABC SBC can limit the bandwidth used by one call to the amount that is necessary in order to prevent denial of service attacks, see Sec. *Traffic Limiting and Shaping* for more details.
- **Traffic monitoring** - the ABC SBC can be used to monitor the amount of media traffic used
- **RTP filtering** - the ABC SBC can filter unknown or not negotiated RTP packets
- **Logging and tracing** - for troubleshooting call audio quality issues, the ABC SBC can be used to get a trace of the traffic including RTP packets
- **Recording** for sake of monitoring, archival or lawful interception - in any of these case the operator must relay RTP packets in order to record the audio.

- **Quality assurance** - especially when protecting a hosted PBX service it is often needed to record incoming calls. To support this feature, an RTP anchoring is needed.
- **WebRTC gateway** - in this mode the ABC SBC must receive the media flows to be able to convert them between plain RTP and secured DTLS-SRTP.

6.9.2 Media Anchoring (RTP Relay)

Media anchoring is activated by applying the **Enable RTP anchoring** action on a call in the A or C rules of either source or destination Call Agent or Realm. This action may be activated several times on a call, however, once activated it can not be deactivated for that call. Executing this action is a prerequisite for many other actions described in this section, they will otherwise not work properly.

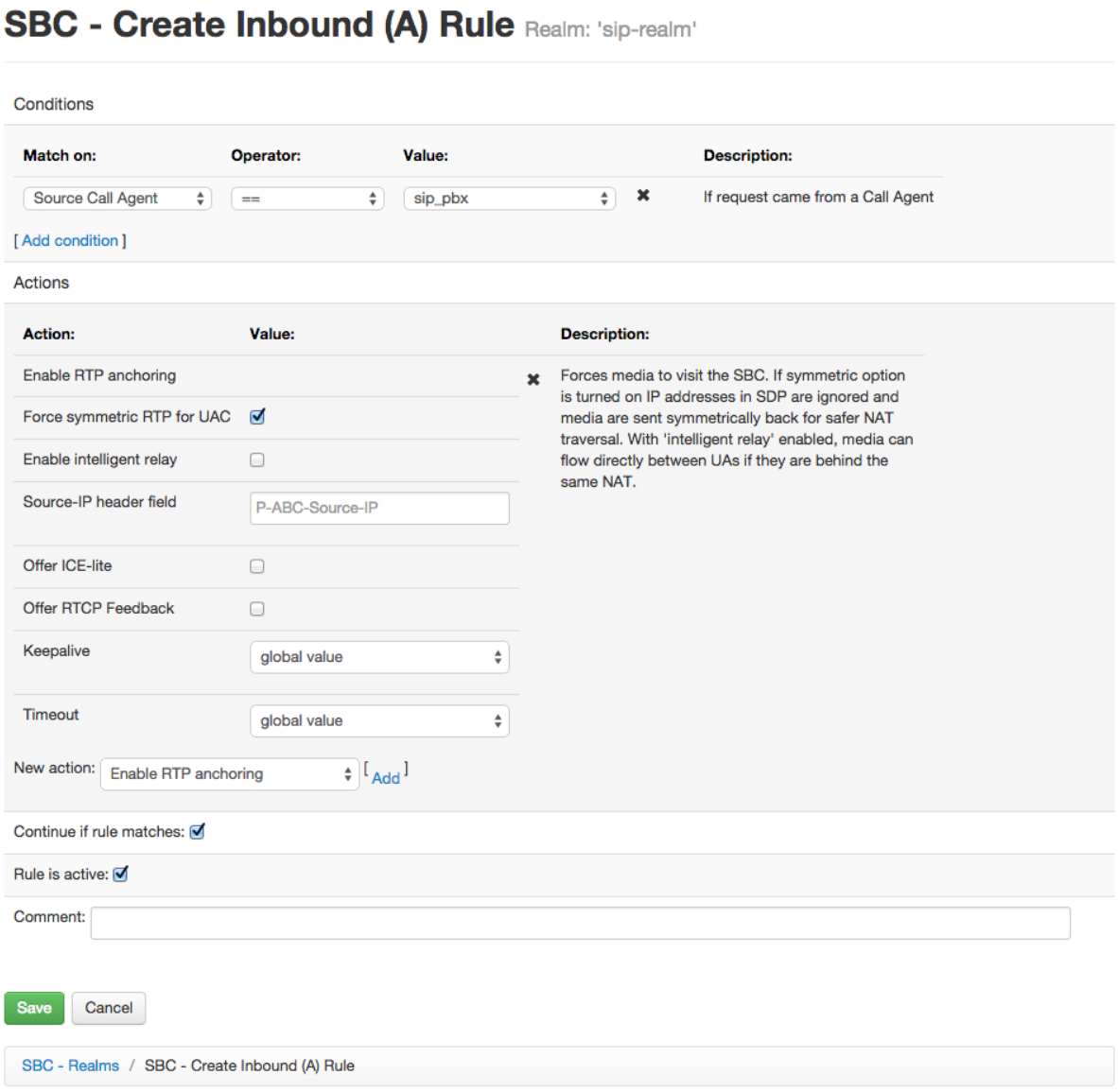


Fig. 35: Media anchoring configuration

The following sub-sections described in more detail respective configuration options of media anchoring.

RTP, RTCP and FAX (T.38) Relay

If media anchoring is activated, both RTP and RTCP packets are relayed for a call. Also, Facsimile over RTP and over UDPTL (T.38) is relayed by the ABC SBC. No configuration step is required, the ABC SBC forwards Fax automatically.

Symmetric RTP Mode and NATs

For User Agents behind a NAT - especially if both user agent are behind NATs - relaying media through the ABC SBC may alone not be enough to accomplish NAT traversal. The problem is the ABC SBC cannot easily determine the public IP address to which to relay RTP media for a SIP phone. The IP address advertised by the User Agents in their SDP payload is non routable.

In a solution here called "Symmetric RTP"[1] (also called Comedia-style[2] RTP handling) the ABC SBC ignores IP address advertised in SDP and learns the IP address and UDP port number of the UA by observing RTP packets coming from the UA. It then starts sending the reverse RTP stream to that address. Symmetric RTP is activated by either one of:

- SIP device, by inserting an "a=direction: active" property in the SDP
- ABC SBC, by enforcing it using the "Force symmetric RTP for UA" option in the "Enable RTP anchoring" action

We recommend to leave this option turned on for all Call Agents in their both inbound and outbound rules. That not only works safely in most cases, but is also more secure in that it prevents use of bogus IP addresses in SDP payloads. Only when a Call Agent is a) known not to be implemented symmetrically AND b) is directly reachable without NATs in between, it makes sense to turn this option for the Call Agent off.

The RTP flows are depicted in Figure *RTP Anchoring with and without Symmetric Mode*. The chart I. shows RTP flows when no media anchoring is engaged. The RTP packets take then the "shortest path" without any SIP intermediary. This flow fails in presence of NATs because telephone's private IP address advertised in SDP is not reachable for the peer device. The chart II shows RTP flows when media anchoring is enabled. The RTP flows from and to a SIP phone are not symmetric, i.e., they are sent from and to different UDP ports as advertised in SDP payload. Like in the chart I, NAT traversal will fail. The chart III shows symmetric RTP that is the safest option for NAT traversal. IP addresses in SDP payload are ignored and the ABC SBC relays media to a telephone to address from which phone's RTP packets come.

Intelligent Relay (Media Path Optimization)

If the ABC SBC handles a call which is originating from the same network that it terminates to, it may be useful to skip media anchoring for that call, in order to save bandwidth and to reduce the total latency introduced. The ABC SBC detects that the caller and the callee are behind the same NAT and is so, bypasses media relay. The test is done by comparing source IP address of incoming INVITE to the intended destination of the request. This algorithm works only if both User Agents are behind the same NAT and there are no intermediary elements between the NAT and the ABC SBC. If this condition doesn't hold, the optimization will fail. That may for example happen if two user agents from behind different NATs speak to the ABC SBC through an intermediary proxy server and appear to the ABC SBC as if they were behind the same IP address. Further, this algorithm does not detect UAs behind a NAT which controls multiple public IPs. Also, the signalling IP address of the callee used for the comparison is projected and does not support domain names.

[1] The name "Symmetric RTP" is derived from the property of a UA that it sends RTP from the address/port combination where it expects to receive RTP at.

[2] The name "Comedia" came from an Internet Draft proposing use of "Connection Oriented Media". The Internet draft draft-ietf-mmusic-sdp-comedia became eventually **RFC 4145**.

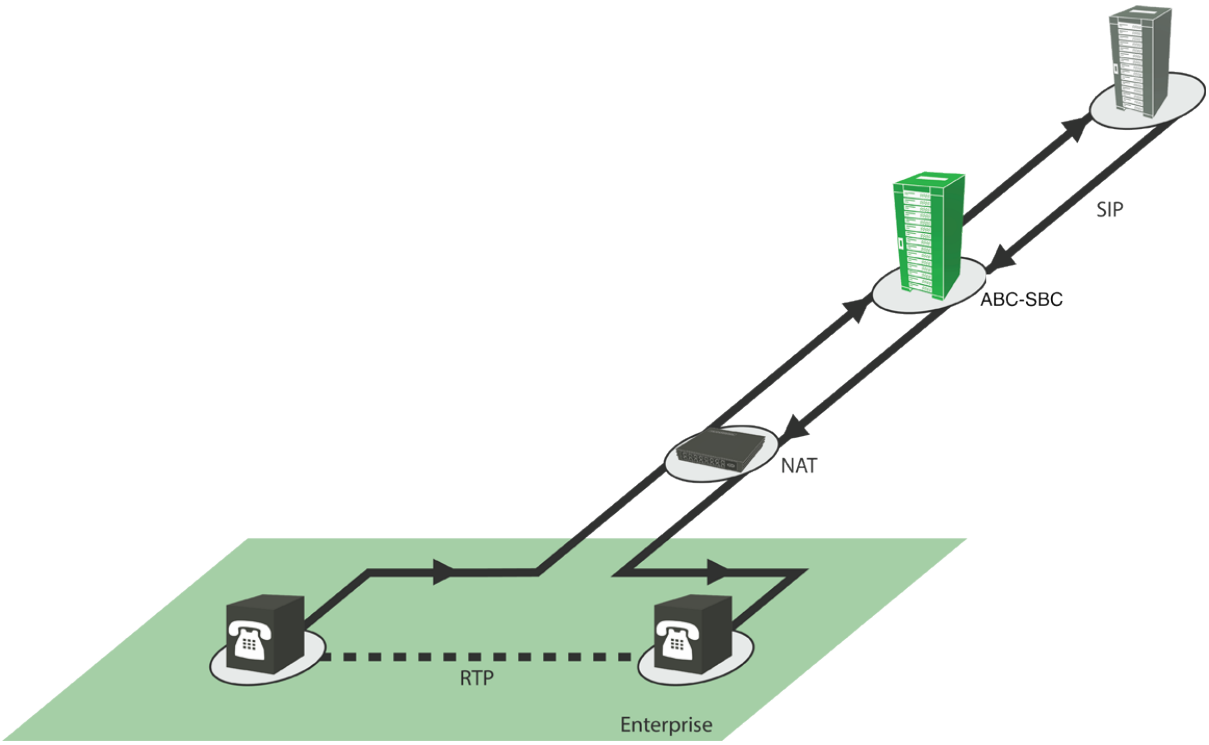


Fig. 36: Intelligent relay

A separate header field, the "Source-IP header field", is used to transport the information about the caller's network through additional proxies in the signalling path. The header name may be configured as a property of the "Enable RTP anchoring" action, so that it can be customised and subsequently filtered out if necessary. This way, the ABC SBC can perform the "same-NAT" test even in scenario shown in Figure *Intelligent relay* and which a call passes the ABC SBC twice. Once on the way in, when source IP address is known but not the final destination , and then on the way out where the destination is already known but the source IP address would be unknown without the additional header-field.

Advanced Anchoring Options

Further media-anchoring options are useful for interaction with advanced clients that use newer protocols: ICE and STUN for NAT traversal, and also additional RTCP feedback for measuring QoS. Such clients are yet rare, however a new interoperability profile for WebRTC clients does actually include all of these. See also Section *SIP-WebRTC Gateway*. Other group of options are those related to keeping calls alive: they make sure that the ABC SBC and its communication peers will properly detect active calls as such, and timely detect calls that ended abruptly.

The following advances options are available:

- *Offer ICE-Lite* – adds ICE-lite (server-side ICE) capability to SDP. This is a must for web-rtc clients that expect their peers to communicate using ICE. WebRTC Call Agents must thus have this option enabled in both A and C rules. It can be also useful for SIP-based User Agents if they support ICE – however generic ABC SBC NAT techniques do not require use of ICE for facilitating NAT traversal.
- *Offer RTCP Feedback* – adds additional RTCP capabilities for sake of finer QoS monitoring than available in traditional RTP implementations. This is mostly useful for WebRTC implementations which include this extension in their interoperability profile.
- *Keepalives* – allows to send keep-alive RTP traffic. This is useful if one side of a call detects and discontinues inactive calls whereas the other side suppresses RTP due to Voice Inactivity Detection or On Hold scenarios. With this option turned on, the calls will not be discontinued.

- *Timeout* – allows call termination when no RTP traffic appears. Useful to eliminate "hanging calls" due to abruptly disconnected SIP devices.

6.9.3 RTP and SRTP Interworking

The ABC SBC can also transform media between "plain-text" RTP and encrypted SRTP. This is particularly useful in SIP/WebRTC interworking scenarios detailed in Section *SIP-WebRTC Gateway*.

The action *Force RTP/SRTP* performs protocol admission in A-rules and protocol conversion in C-rules. When placed in A-rules, it only permits calls corresponding to the requested protocol, the calls will be rejected otherwise. When the action is placed in C-rules, it converts media to the chosen protocol. If the chosen protocol is SRTP, the keying protocol must be also chosen: DTLS or SDES. When the destination is a WebRTC client, the keying protocol must be DTLS since spring 2014.

SRTP with RTP fallback ("SRTP fallback to nonsecure RTP") is a method of optional SRTP (opportunistic encryption) where the offerer sends an SRTP offer, but the answerer can fall back to RTP in case SRTP is not supported. This means that, contrary to SDP offer-answer requirements of RFC3264, the transport of the answer can be different to the offer: it can be RTP/AVP(F) when the offer is RTP/SAVP(F). This Cisco-specific method also adds a Supported tag "X-cisco-srtp-fallback". Whether SRTP or RTP is used can be renegotiated at every Offer-Answer exchange (e.g. re-Invite). This fallback method does not try to re-negotiate non-secure RTP if a 488 is received.

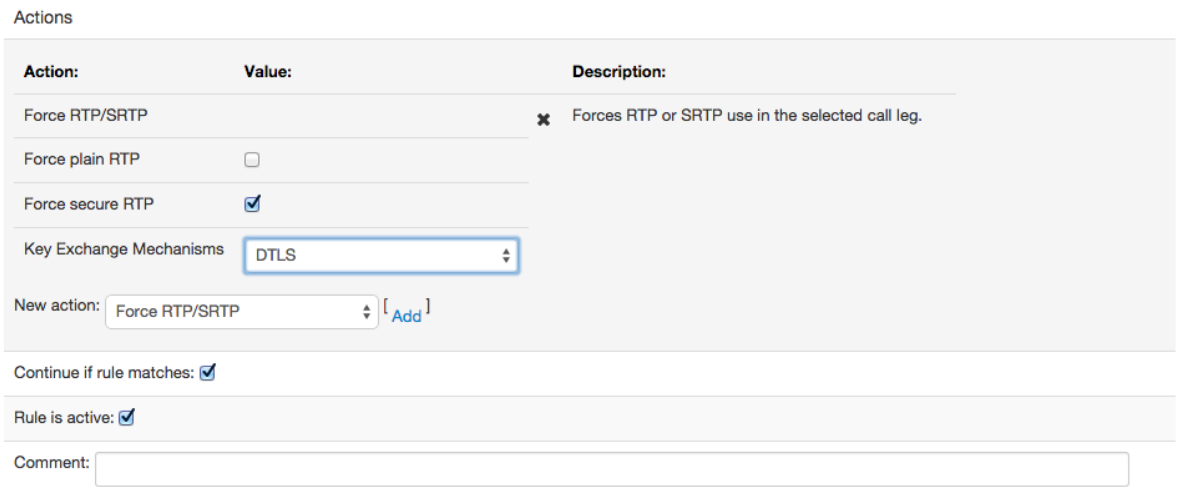


Fig. 37: Enforce SRTP

6.9.4 SRTP End to End encryption

The action *End to End encryption* provides the capabilities to stay in the media path while not interfering in the SRTP negotiation. The SRTP key is negotiated by both peers without any intervention of the SBC, which is not able to encrypt/decrypt the media if this action is enabled. The RTP or SRTP packets are then just relayed as-is.

6.9.5 Transcoding

To enable broader interoperability, the ABC SBC can transcode between different codecs, that is it will decode incoming RTP packets and encode RTP packets into a different codec. The ABC SBC currently supports transcoding for audio only, there is currently no support for transcoding video streams.

For transcoding to be available in the ABC SBC, the operator needs to get and install the proper license for the media processing package, see Sec. *Installation Procedure* for details. Also, for some codecs, patent licenses need to be acquired separately. For some codecs, special software packages need to be installed, please contact FRAFOS support if in doubt.

The FRAFOS ABC SBC supports software based transcoding. Transcoding adds non-negligible processing power requirements to the SBC hardware, see Sec. *SBC Dimensioning and Performance Tuning* for details.

Depending on the codecs used, transcoding also reduces voice fidelity, especially if transcoding is applied a multiple times on the path of the call.

The action **Activate transcoding** takes as parameter a comma-separated list of codecs which are added to the SDP offer, if not present in the original offer. The codecs listed here must be supported by the ABC SBC . If the other party accepts one of these, the media stream is transcoded. Both (or, all) codecs which the ABC SBC should transcode between need to be added to the list of transcoding codecs.

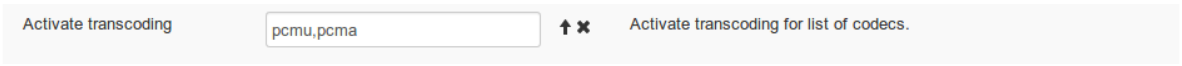


Fig. 38: Activate transcoding

For example, if "pcmu,pcma" is configured as transcoding codecs, and the caller offers only PCMU and the called party PCMA, the ABC SBC transcodes from PCMU at one side to PCMA at the other side and the other way around. If both sides happen to support the same set of codecs then transcoding will not be needed and will not be used.

6.9.6 Audio Recording

Recording may be useful for a variety of purposes: most often for archival, monitoring and legal interception. If a call is selected for recording, the ABC SBC collects audio and stores it in a WAV file. Each direction is stored in one channel, the file is stored with sampling rate 8kHz, two bytes per sample (PCM), two channels. To allow recording, media anchoring must be turned on. Recording works only if supported codecs are used.

Recording is activated by the action *Activate audio recording* that takes a comma-separated destinations as parameter.

The destination may be a filename, a HTTP server to which the WAV file is uploaded using the PUT method or a SIP URI if the call recording is to be outsourced to a SIPREC call recording server. If a SIP URI is used, only one is supported and should not be entered as a list.

The call recording action supports two more parameters allowing for start and stop announcements. These announcements are played when the recording starts or stops. Please note that the stop announcements can only be played if SIPREC is used, and the call recording server (SRS) does stop the recording before the call has been ended.

Replacement expressions can be used to provide easier identification of the system. USE CAUTION when devising the filename: filename conflicts will result in different sessions overwriting each other's WAV file. If no filename is included, the ABC SBC uses its own ephemeral filename. Filenames are made relative to the directory /data/recordings to make sure that the recording doesn't interfere with the filesystem.

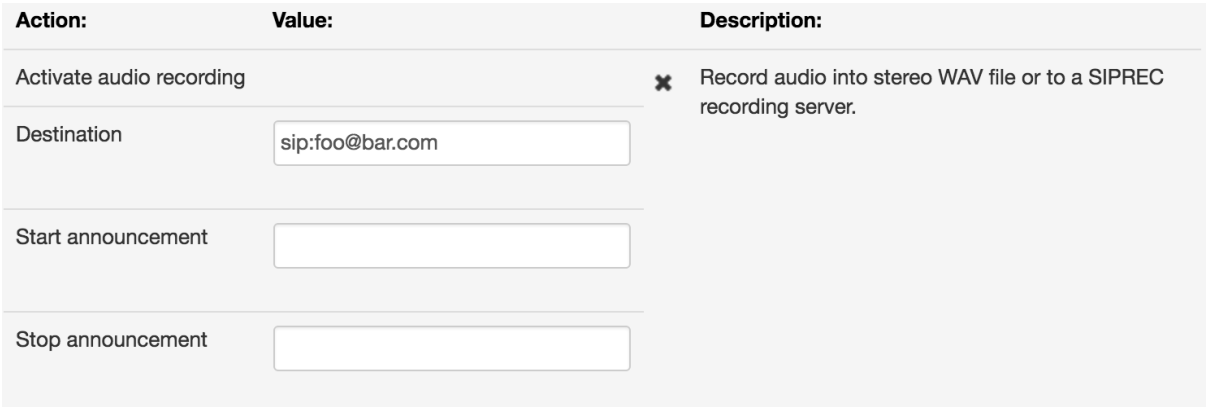


Fig. 39: Activate Audio Recording

When recording and generating the WAV files completes, an event is produced.

Note that to avoid premature deletion of important archival data, the system does not delete any audio files and keeps them stored. This may potentially exhaust the disk space. Consult professional services if you need help on managing audio archives.

SIPREC specific options

When a SIPREC is used for audio recording, a set of specific options can be configured.

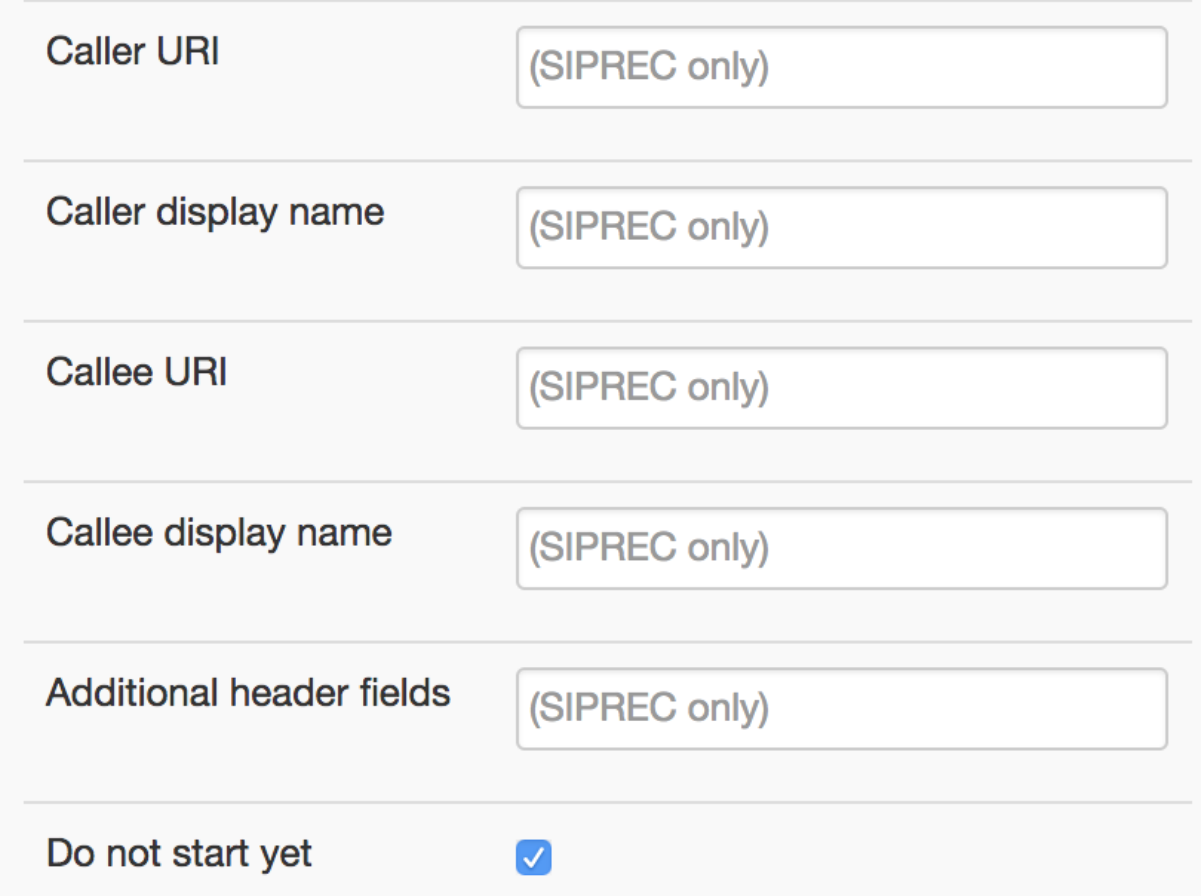


Fig. 40: SIPREC options

The fields related to caller and callee determine the values transmitted to the recording server within the SIPREC metadata. The Caller & Callee URIs are used to set the participants' `nameID aor` tags, while Caller/Callee display names are used to set the `name` tags.

The field "Additional header fields" can be used to add header(s) to the SIP INVITE message sent to the SIPREC server.

The last option "Do not start yet" allows to delay the start announcement until the SIPREC server signals it has started the recording. This allows for notifying the user properly. Please note that the media streams are transmitted during the complete call to the recording server, even if this feature is used.

Since ABC SBC 4.5 it is possible to configure SIP timers towards SIPREC server. With appropriate values this may help to speed up error detection. See *SEMS Parameters* for configurable options.

6.9.7 Playing Audio Announcements

Often it is practical to inform caller about an error by an audio announcement rather than a numerical SIP code. The ABC SBC can play audio files on several different occasions for each of which there is an appropriate action:

- "Refuse call with audio prompt" plays an audio file immediately on receipt of an INVITE
- "Play Prompt on Final Response" plays an audio file on an unsuccessful call attempts
- "Generate Ring-Back Tone" plays an audio file instead of the default ringing tone
- "Activate Music On Hold" plays an audio file when a party chooses to put a call on hold

The action "Refuse call with audio prompt" plays a prerecorded audio WAV file immediately on receipt of a SIP INVITE. It is typically used to decline a call using a pre-recorded message. This action plays a prerecorded WAV audio file and terminates the call. It takes the following parameters:

- filename of the announcement relative to the global configuration option "Prompts/Base Directory". The filename must refer to an existing file which has been uploaded to the prompts directory by administrator.
- a checkbox specifying whether the announcement shall be played as early media or establish a regular call
- a checkbox specifying whether the announcement shall be played once or in a loop
- SIP response code, phrase and header-fields to be used for terminating early media announcement (unused if a regular call is established)
- also instead of playing a pre-recorded WAV file, a beep tone can be generated. To turn it on, activate the "generate ringtone" checkbox and describe the tone length, on-off periods and frequencies.

Fig. 41: Example Action for Playing an Announcement

The same effect can be achieved for SIP calls that failed downstream. The action **Play Prompt on Final Response** plays an announcement for call attempts that failed downstream due to one of the listed failure codes. The announcement can be played as a regular call or as early media, in which case a specified SIP response code will conclude the announcement.

Fig. 42: Example Action for Playing an Announcement on Receipt of a 404 Response

Also it may be useful to play specific tones or audio when called party's telephone is ringing. To enable this functionality, use the action "Generate Ring-Back Tone" and either define a tone or reference to an audio file to be played instead of the default ringing tone. The screenshot in Figure *Example Action for Playing an Audio File during Ringing* is showing such a configuration that plays a predefined audio file during the ringing phase after the receipt of the UAS's 180 response. Alternatively one could have played a predefined dual-frequency tone.

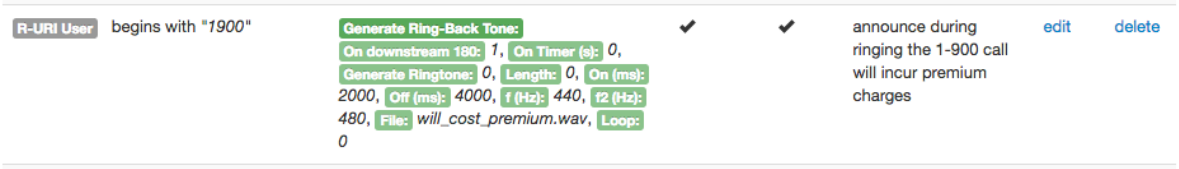


Fig. 43: Example Action for Playing an Audio File during Ringing

Similarly it is possible to play an audio file whenever a party chooses to put a call on hold. The action takes several parameters that allow it to define how the on-hold status is signaled to the other call party and if the audio file is played once or in a loop.

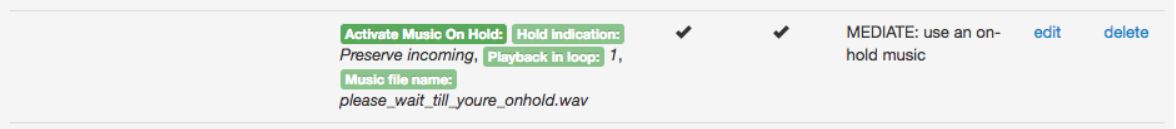


Fig. 44: Example Action to Activate an Audio File when a Call is Put on Hold

6.9.8 Onboard Conferencing

To accommodate smaller-scale dial-in conferences without need for an external conference bridge, the ABC SBC can mix audio calls. To enable a conference, place a "join meet-me conference" action in A rules. The action's parameters allow to specify which conference an incoming call shall join: either by two-stage DTMF dialing, or by a "hard-wired" conference ID, or by conference ID determined using a replacement expression.

If the room is entered via keypad (DTMF), then some more parameters control how that is done: A minimal length of the room can be set, and also some unacceptable room numbers (e.g., too simple, can be guessed). Once the room is entered via the keypad, a prefix can be prepended to it: This way, separate 'namespaces' of conference IDs can be used, e.g. if the same SBC is connected to two different networks which should never share a conference room, but in both of them the room ID should be entered via the keypad.

The room entered via keypad can also be split at a specific position into room ID and participant ID by using the "Split room and participant ID" setting. This way, a web interface can send out invitations with individual PINs and later identify the callers by their participant ID, while the different participants still hear each other in the room.

The following example configuration shows a conference bridge configured to serve calls from native SIP devices, SIP-based PSTN gateways and WebRTC browsers. SIP devices and WebRTC browsers are handled the same way: userpart of request URI ($rU) identifies the conference a caller is joining. Calls from the PSTN call-agents however are prompted to type in the conference id when dialing in.

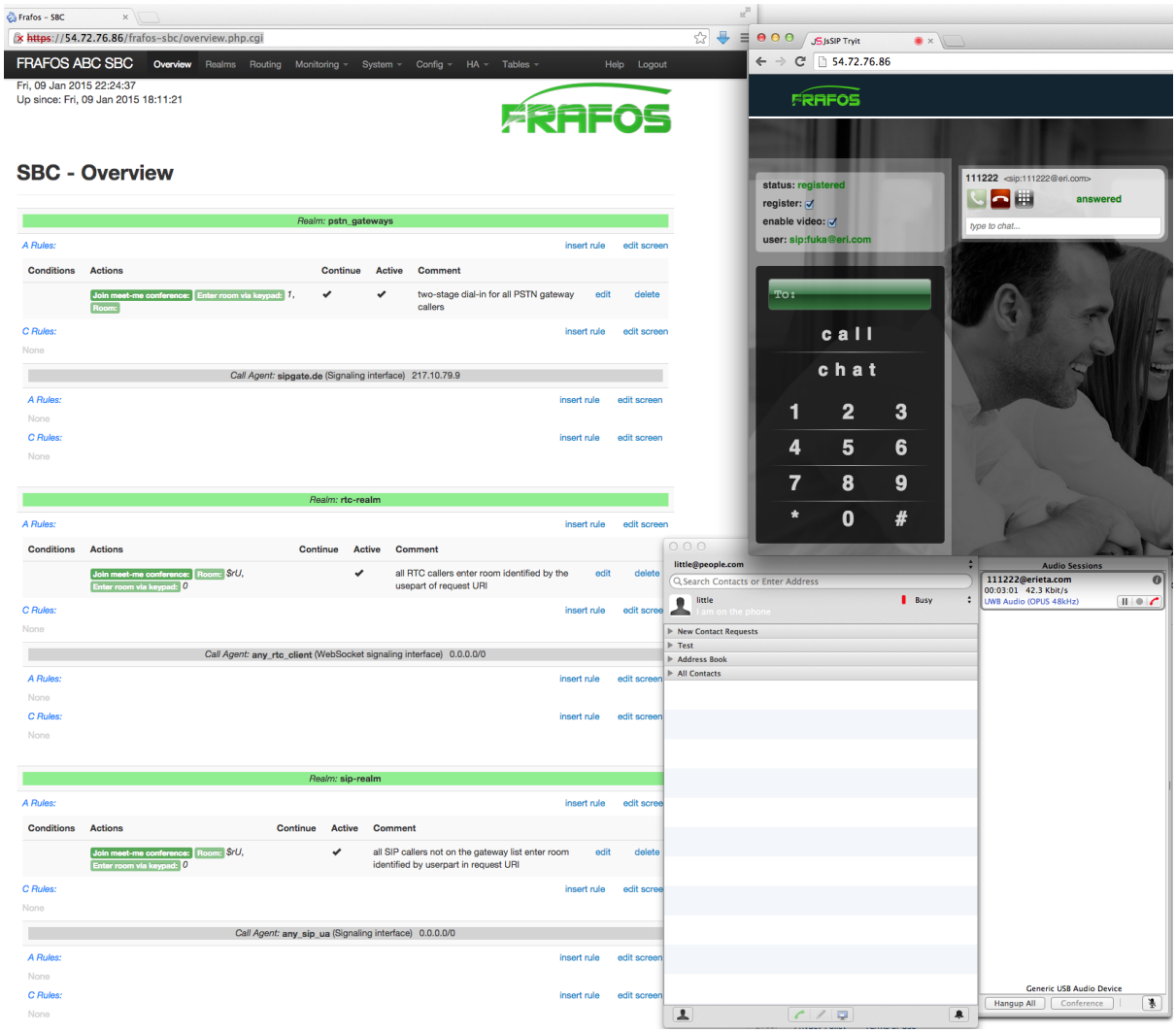


Fig. 45: Example Screenshot: Conference Bridge configuration

Note that the default WAV announcement files (like conference ID prompt, or "you're welcome" message) are stored on the system in the directory /usr/lib/sems/audio/webconference and can only be changed through Command Line Interface. Also note that locally processed onboard conferencing calls do not appear in the list of live calls (see Section *Live Calls*) in which only relayed calls are represented.

The status of the conference bridge can be inspected using CLI as shown in the following example

```
[root@ec2-54-154-137-127 ~]# sems-stats -c "DI webconference listRooms verysecret"
sending 'DI webconference listRooms verysecret\n' to 127.0.0.1:5040
received:
[200, ['1234'], 'Server: Sip Express Media Server (3.1.1-79-c86706a-417e607-87219e5
(x86_64/linux)) calls: 1 active/0 total/0 connect/0 min']
[root@ec2-54-154-137-127 ~]# sems-stats -c "DI webconference roomInfo 1234 4447"
sending 'DI webconference roomInfo 1234 4447\n' to 127.0.0.1:5040
received:
[0, 'OK', [['020D64C4-54E33359000B632C-E54DD700', '"homer" <sip:homer@simpson.org>
↪',
3, 'direct access: entered', 0, '']],
'Server: Sip Express Media Server (3.1.1-79-c86706a-417e607-87219e5
(x86_64/linux)) calls: 1 active/0 total/0 connect/0 min']
[root@ec2-54-154-137-127 ~]#
```

Conferencing room pin protected

Conference room may also be protected in a form of a security PIN. That security PIN is asked to each participant before joining a room.

To use this option, please enable the "Room is PIN protected" option of the "Join meet-me conference" action. PIN management is subject to 3 possible options :

- First user to join is prompted to set the security PIN

To do so, please enable the "Room is PIN protected" option and leave the "PIN" field empty.

- PIN is set via action rule

To do so, please enable the "Room is PIN protected" option and set the "PIN" field to the desired security PIN.

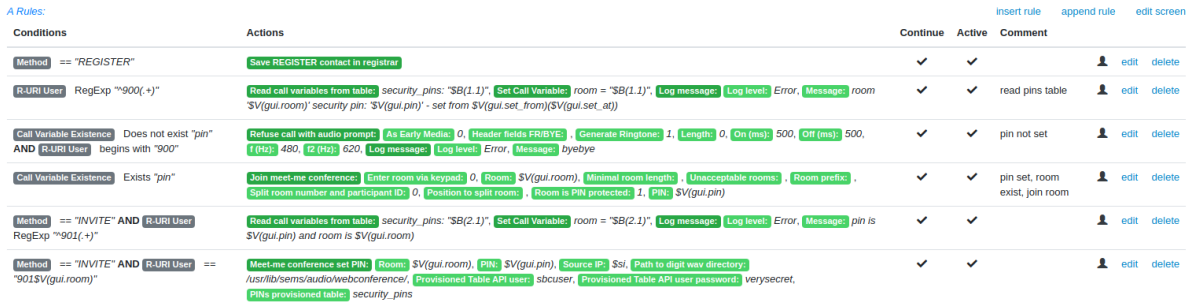
- PIN is set via another action

User may use the "Meet-me conference set PIN" action to set and persist a security PIN into a specific provisioned table.

In the following scenario, user is required to first set the pin of a room before accessing it. We request the user dial the *9011234* so the security pin of the room *1234* may be set. He's then able to dial the *9001234*, where he'll be prompted for the security PIN of the room *1234* before being able to join.

Please start by creating a provisioned table of type "pins". Use the "Meet-me conference set PIN" action to set the PIN of a room and persist that value into the provisioned table. You may then fetch the PIN value from the table and use it as call variable (see following rules screenshot)

See the following rule configuration as example :



6.10 NAT Traversal

SIP devices behind Network Address Translators (NATs) cannot reach other SIP devices reliably. The root reason is SIP protocol advertises SIP device's IP addresses in several places in the protocol: *Contact* and *Via* header fields SDP *c* line. These addresses are non routable once they cross a NAT device and break signaling. The ABC SBC is designed to assist the to facilitate NAT traversal for SIP devices by several techniques: it centers itself in the middle of communication path, sends signaling and media reversely to where it came from even in violation of the SIP standard, replaces private IP non routable addresses with its own and keeps all bindings alive.

As depicted in Fig. *SBC and NAT traversal*, when an INVITE request traverses a NAT then only the IP addresses in the IP header will be changed. Any IP addresses included in the message itself, e.g., *Contact*, SDP *c* line, will still reference the private IP address of the caller. As the callee would use the information in the *Contact* header for replying back to the caller and send media packets to the address in the SDP the call establishment will fail.

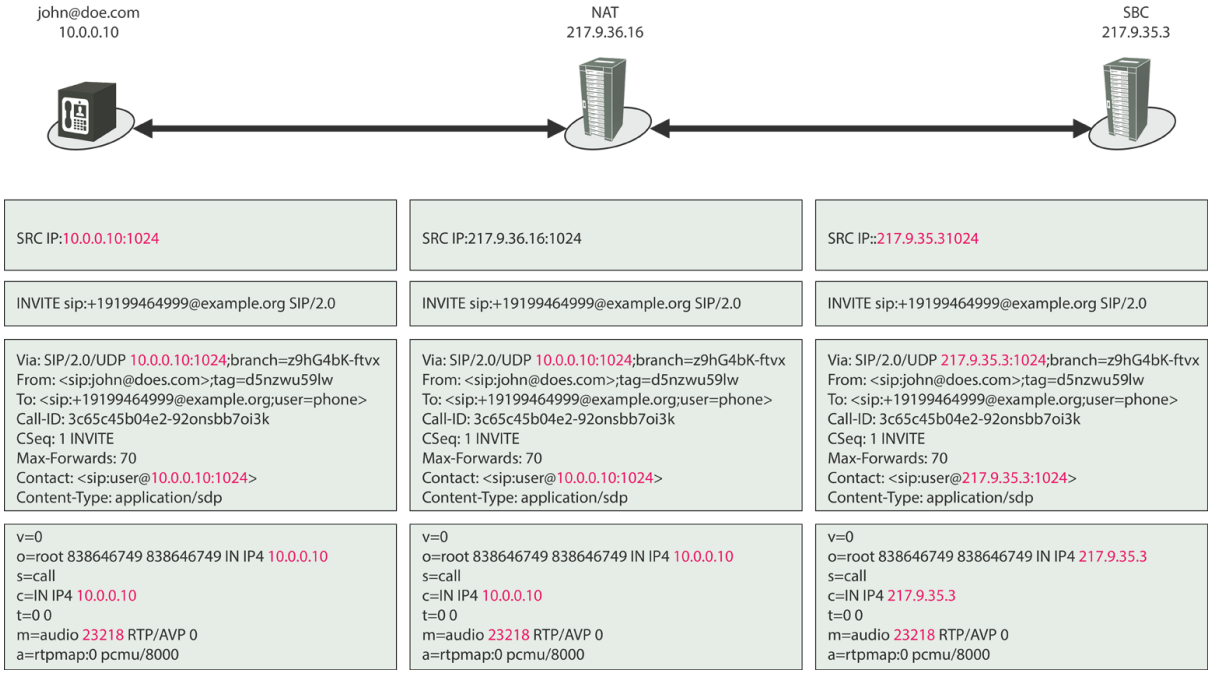


Fig. 46: SBC and NAT traversal

In the context of NATs, there are basically three different possibilities with respect to the network topology that will influence the possible measures that can be taken by the ABC SBC to deal with those NATs:

- **Far end NAT:** This is the most common case in public SIP service scenarios. The SBC is located on the public Interned and the end-devices access the SBC from behind NATs. The SBC must facilitate the NAT traversal for the end-devices: it must accomplish RTP traversal, SIP traversal and registration off-load.
- **SBC is placed on the NAT:**. This is the most common case in enterprise deployments in which the SBC acts as firewall between a private network with SIP telephones and PBXes, and the outside network. It has at least one signalling and media interface inside and one outside the NAT. SIP signaling is handled natively without any additional configuration. However, it is necessary to enable RTP anchoring (relay) so that the media payload can flow from between the otherwise non routable networks.
- **Near end NAT:** here, the SBC is placed right behind the NAT and a port forwarding is configured from the NAT to the SBC. This is considered by the ABC SBC as a special case of the previous configuration. In this case, it is perfectly sufficient to configure the signalling and media interfaces with the public IP address on the outside of the NAT. It is also necessary that the configured port range on the media interface corresponds to the forwarded ports for media transport, as no port translation is supported at this place.

In order to enable users behind a NAT to be reachable the ABC SBC needs to perform the following tasks.

- Detect if a SIP user is behind a NAT. This allows to eliminate expensive NAT traversal facilitation for users who do not need it. The test is performed by the condition **NAT** in inbound rules.
- Fix outgoing calls: The ABC SBC must fix SIP INVITEs from users behind NATs so that subsequent SIP messages coming back will cross the NATs successfully. Particularly, the SBC fixes Contact header-field and stores NAT information in dialog context. This functionality must be enabled in inbound rules using the **Enable Dialog NAT Handling** action.
- Fix incoming calls: The ABC SBC must deliver incoming INVITEs for a user behind a NAT through the NAT devices. This is only possible if the NAT devices keep the UDP or TCP binding open over which the user registered. Otherwise the user becomes unreachable once the binding expires. Therefore the ABC SBC pays great attention to the SIP registration process: It can force more frequent registrations to keep the bindings alive and it also keeps source address from which the registration came. Subsequent requests for the registered client are forwarded to the address (as opposed to the private IP address advertised in Contact header-field). To enable this functionality the actions **REGISTER throttling** and **Enable REGISTER caching** must be applied to REGISTER messages, and the action **Retarget R-URI from cache**, with the

enable NAT handling option turned on must be applied to calls towards the client. See section *Registration Caching and Handling* for more details.

- Media anchoring: The ABC SBC redirects RTP stream to itself and sends symmetrically one's party RTP media to where the RTP media from the other party came from. This symmetric mode of operation overrides SIP signaling but works more reliable, because it is better compatible with how most NAT devices work. The downside of this approach is the extra bandwidth consumption on the ABC SBC and increased RTP latency. Media anchoring is enabled by the action **Enable RTP Anchoring**. "Force symmetric option" can be turned on and off for UACs in inbound and UAS in outbound rules. Media handling and RTP anchoring ABC SBC is described in more detail in Sec. *Media Handling*.

In summary the following conditions and actions are used to configure NAT traversal:

- **NAT** condition - check if the first Via address is or is not behind NAT. This checks if the first Via address is the same as the IP address that the SIP message was received from.
- **Enable dialog NAT handling** action - force all subsequent in-dialog messages to be sent to IP/port from which the dialog-initiating request came.
- **Enable RTP anchoring** action - force RTP from a SIP user to be sent through the ABC SBC. The **Force-symmetric RTP** option forces media from the other side to be sent reversely to where the user's media came from. Turn it off only if a Call Agent is known to reject symmetric media.
- **Enable REGISTER caching** must be applied to REGISTER messages for retargetting to function. See section *Registration Caching and Handling* for more details.
- **Retarget R-URI from cache (alias)** action. This action makes sure that INVITEs coming to a registered client will reach it by sending them to the transport address from which a REGISTER came. The options **Enable NAT handling** and **Enable sticky transport** should be turned on.

The example in the following subsection shows how to place the respective actions in the ABC SBC rulebase.

Note: In an actual deployment, the specific topology needs to be considered carfuly. For example, if SIP passes the SBC twice, RTP could without precaution be also anchored twice resulting in unnecessary performance degradation. That's because the SBC recognizes inbound and outbound call legs as two separate calls.

6.10.1 NAT Traversal Configuration Example

This example shows how to put all the NAT-faciliating actions in a consistent rule-base. This example is based on the "far-end" NAT traversal topology as shown in the Figure *NAT Traversal Example: Network Topology*. In this topology, the ABC SBC is multihomed: it connects to the public Internet with one interface and to a private network with the other interface. Both networks are not mutually routable, the ABC SBC connects them on application layer. SIP telephones are on the public side behind NATs, service provider infrastructure including a SIP registrar, proxy, media server and PSTN gateway are located in the private network.

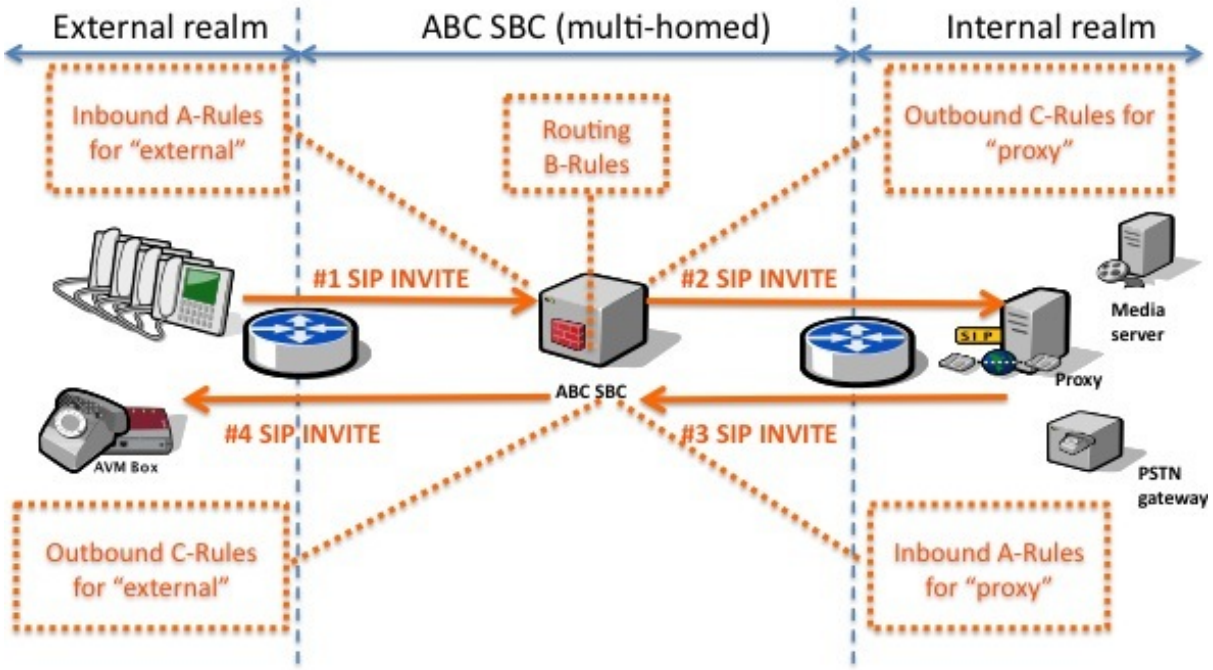


Fig. 47: NAT Traversal Example: Network Topology

The first preparatory step to be is handling telephone's outgoing SIP messages coming from the external realm. That is done in the external realm's A rules: Dialog-initiating requests must be fixed in a way that reverse messages will follow the same path. REGISTERs must be stored along with the transport address from which they came. Frequent re-registration must be enforced to keep NAT-bindings alive. Eventually RTP anchoring must be enabled. The configuration fragment is shown in the Figure *A-rules for traffic coming from outside*.

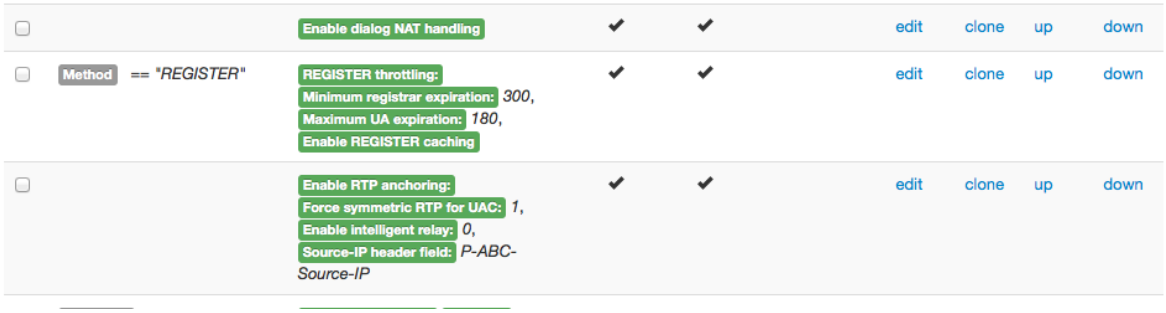


Fig. 48: A-rules for traffic coming from outside

When requests pass the SIP proxy and come again from the inside network to the SBC, the addresses in them must be reverted to the form used initially by SIP User Agent. The configuration fragment is shown in Figure *A-rules for traffic coming from inside*.

Fig. 49: A-rules for traffic coming from inside

Eventually RTP anchoring is turned in in C-rules for calls going to the public network, as shown in Figure *C-rules for traffic leaving for outside*.

Fig. 50: C-rules for traffic leaving for outside

6.11 Registration Caching and Handling

ABC SBC's registration cache mediates the registration flow between SIP User Agents and SIP registrars. It keeps track of SIP User Agent contacts and shields SIP registrar from overload. It also facilitates NAT traversal. In case a user agent is located behind a NAT it will use a private IP address as its contact address in the *Contact* header field in REGISTER messages. This non routable address would be useless for anyone trying to contact the user agent from the public Internet.

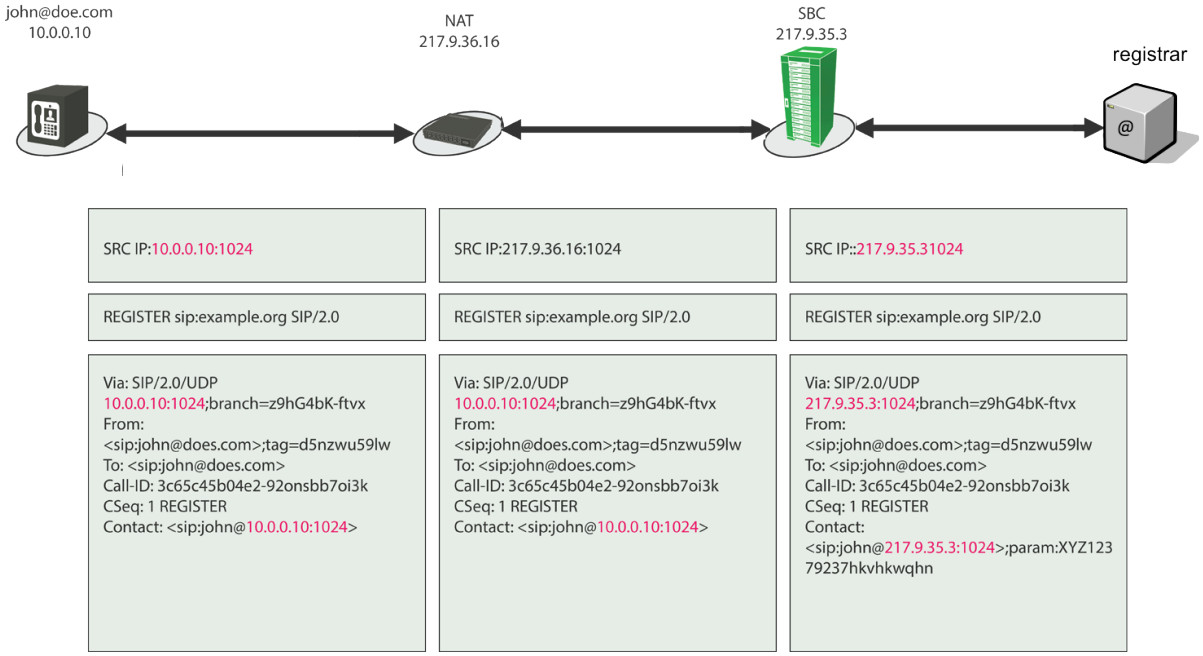


Fig. 51: SBC and NAT traversal: Registration handling

In order for a SIP User Agent to be reachable the ABC SBC will manipulate its registration information. The SBC remembers the User Agent's transport address and replaces the information in the *Contact* header with its own IP address before forwarding downstream, see Fig. *SBC and NAT traversal: Registration handling*. This is the information that is then registered at the registrar. Calls destined to the user will then be directed to the SBC. The SBC then forwards the calls using previously stored information about the User Agent' transport address and initial unmodified contacts.

The registration cache implements the following functions:

- **Contact fixing:** SIP contacts of User Agents behind NATs include private IP addresses which are not routable from the public Internet. Therefore the ABC SBC rewrites the IP address in the Contacts with its own and holds the original Contact in its cache. If the ABC SBC connects to multiple networks using multiple IP addresses, the IP address is used which is associated with the interface over which the REGISTER request is forwarded. When later incoming requests towards the User Agent reach the ABC SBC, the ABC SBC restores the original address.
- **Keeping NAT-bindings alive**. If periodic request-response traffic was not crossing the NAT behind which the User Agent is located, the NAT address binding would expire and the client would become unreachable. Therefore the ABC SBC steers User-Agents to re-register often.

- **Registration off-loading**. Various circumstances can cause substantial registration load on the server: most often it is self-inflicted by the keep-alive functionality, but it may be also on the occasion of a registration storm caused by a router outage, broken client or Denial of Service attack. The ABC SBC fends off such overload by using high-performance in-memory registration cache that serves upstream registrations at high-rate, handles them locally, and propagates them down-stream at a substantially reduced rate. That's the case if the registrations were to create new bindings, deleting existing ones or if they were to expire downstream. The propagated registration changes become effective on the ABC SBC only if confirmed by the downstream server. If a registration expires without being refreshed the ABC SBC issues a reg-expired event.

The following Subsection, *Registration Handling Configuration Options*, documents the specific actions that implement the cache functionality. Note that the procedures described here refer to individual URI registration as envisioned in the **RFC 3261**. Provisioning of bulk registration for PBXs as specified in the **RFC 6140** is described in the Section *Table Example: Bulk Registration*.

6.11.1 Registration Handling Configuration Options

The ABC SBC can handle SIP registrations in two ways, either caching them locally and forwarding them to a downstream registrar, or acting itself as a SIP registrar.

If the ABC SBC fronts a registrar, the action **Enable REGISTER caching** is applied on incoming REGISTERs from a User Agent to cache and translate its Contacts. On the reverse path towards the User Agent, the action **Retarget R-URI from cache(alias)** restores the original Contacts.

- **Enable REGISTER caching** - cache contacts from REGISTER messages before forwarding, create an alias and replace the *Contact* with *alias@SBC_IP:SBC_PORT;contact_parameters*. This method should only be applied to REGISTER messages to be forwarded to a registrar. This action has effects only on REGISTER requests, see Fig. *Enable Register caching*.

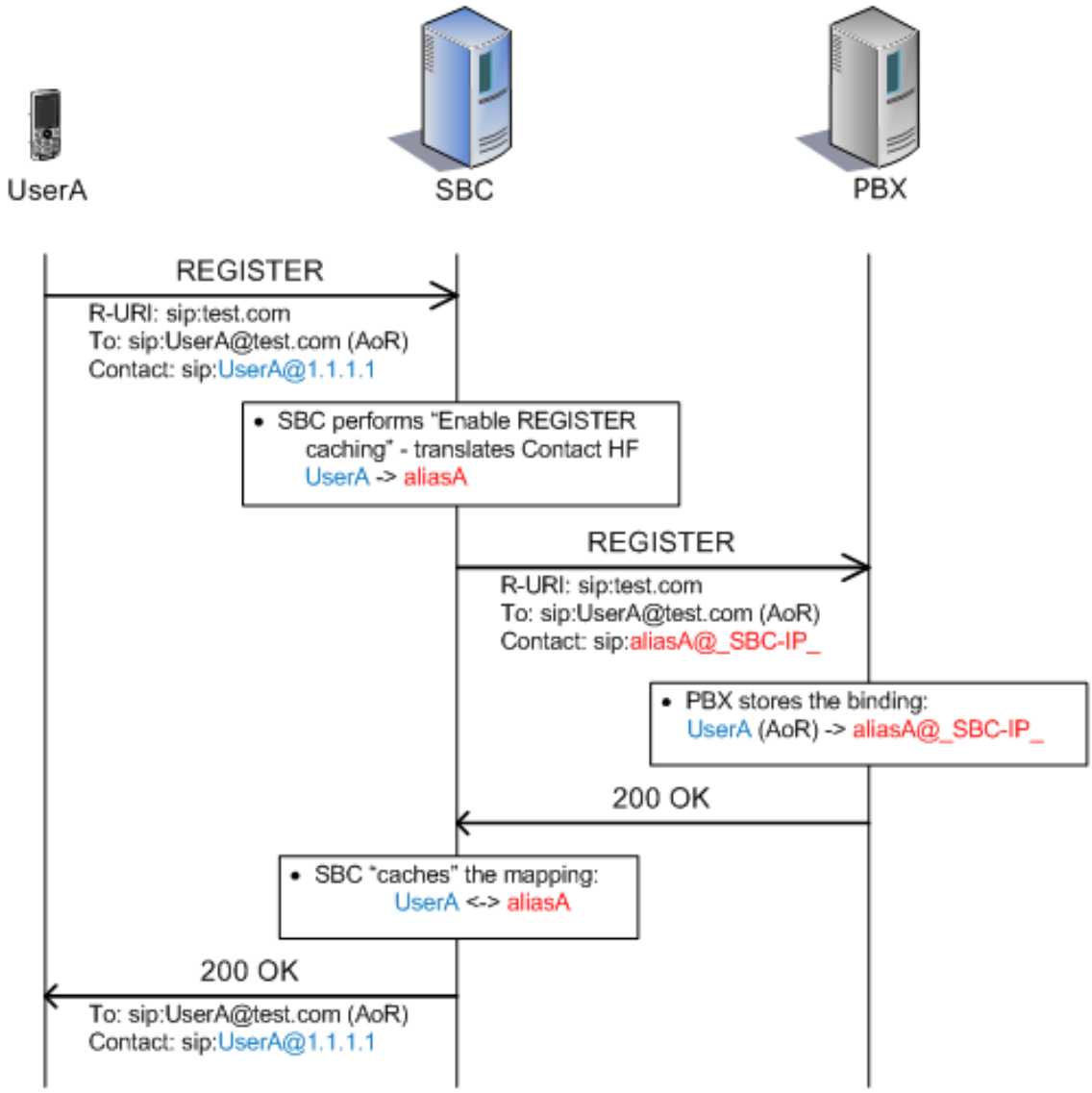


Fig. 52: Enable Register caching

- **Retarget R-URI from cache (alias)** - Look up cached contact under alias and rewrite the request URI with it. Apply this action to messages sent to clients whose registration were cached previously using the "Enable REGISTER caching" action. This scenario is depicted in Fig. *Retarget R-URI from cache*. When an INVITE arrives to a user that has previously registered its contact information (*UserB*) the ABC SBC will forward the INVITE to the PBX which acts in this case as the SIP proxy and Registrar. The PBX will look for the registration information of *UserB* which in this case are *aliasB@_SBC-IP_* and use this information for routing the request. When the INVITE with the Request URI set to *aliasB@_SBC-IP_* arrives at the SBC, the ABC SBC will check its registration cache and retrieve the actual contact information of the user, namely *UserB@2.2.2.2* and use this information as the Request URI and forward the message to this address.
- Parameters: **Enable NAT handling**:source IP and port of the REGISTER request; **Enable sticky transport**: use the same interface and transport over which the REGISTER was received.

Note: if no matching entry is found in the cache, the ABC SBC returns a 404 SIP response.

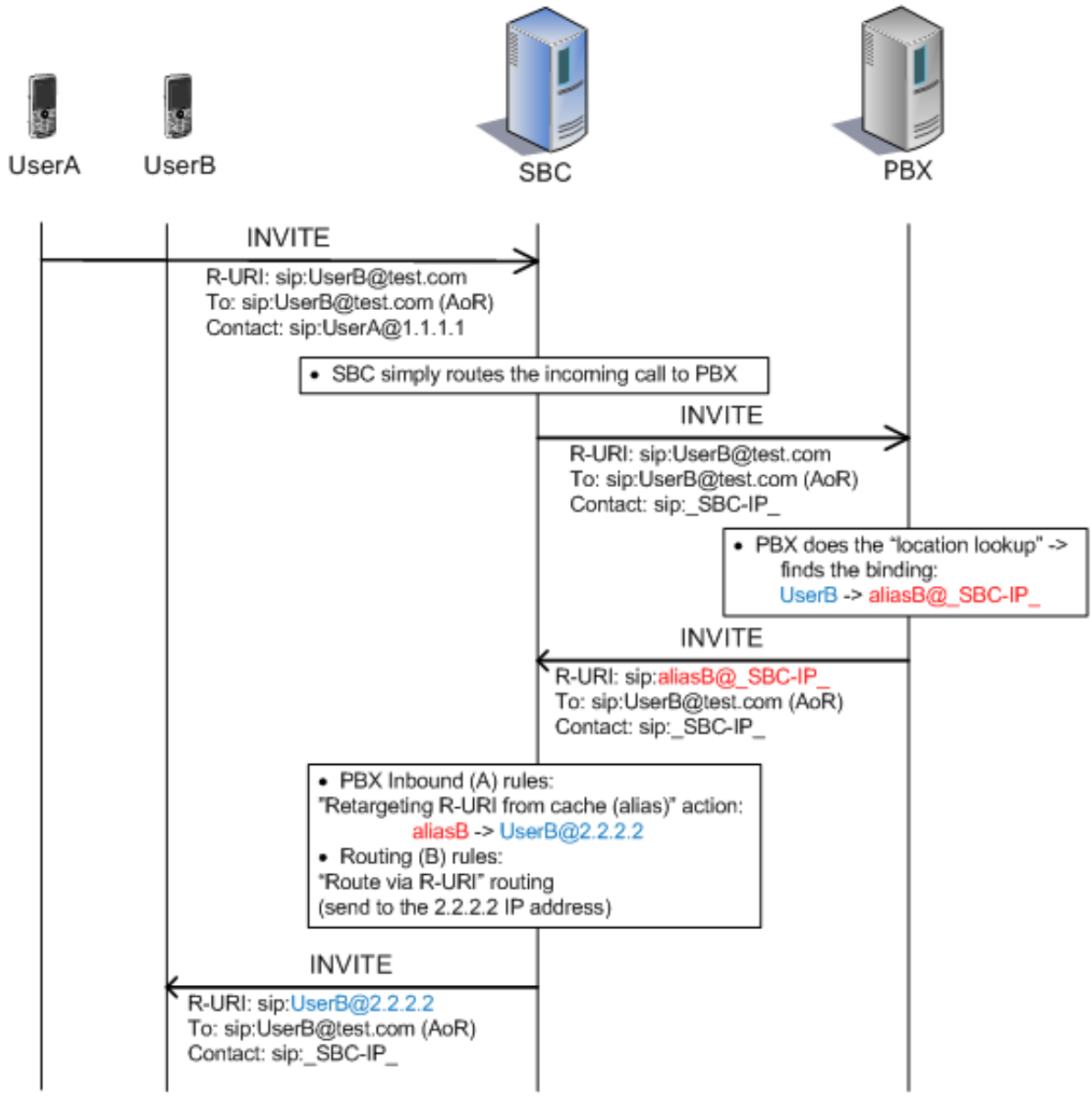


Fig. 53: Retarget R-URI from cache

If the ABC SBC is used as registrar, the two following actions are used instead: **Save REGISTER contract** for REGISTERs and **Restore contact** for incoming requests towards the User Agent.

- **Save REGISTER contact in registrar** - act as local registrar by saving the contact and replying with a 200 response.
- **Restore contact from registrar** - use contact stored within internal registrar

6.11.2 Registrar off-load

Registration throttling can be used in both built-in registrar and registrar-cache modes. The action **REGISTER thottling** enforces high re-registration rate towards User Agent Client and a reduced rate towards registrar. The high upstream rate serves the purpose of preserving connectivity by keeping address binding along the SIP path alive. The reduced rate on the downstream side makes sure that the connectivity traffic doesn't overload the downstream registrar. The **REGISTER throttling** action must precede any other REGISTER-processing action, otherwise its load-reducing function will take no effect:

- **REGISTER throttling** - force SIP user-agents to shorten re-registration period while propagating the REGISTERs upstream to registrar at longer intervals. This is useful to keep NAT bindings open without imposing the refreshing load on registrar.
- Parameters: **Minimum registrar expiration**: expiration time used in direction to registrar. **Maximum UA expiration**: maximum expiration time in direction to User Agent Client.

Note that these two parameters have vast impact on the volume of SIP traffic: they steer the registration rates towards upstream SIP client and downstream SIP registrar.

The **Maximum UA expiration** "knob" steers the SIP traffic rate between upstream SIP client and the ABC SBC. It suggests to the UA at which time interval it shall re-register. The lower value is enforced, the higher the registration rate will be. The SIP standard suggests one registration per hour which is not good enough to keep NAT bindings alive. Forcing the re-registration interval down to 180 seconds will cover a satisfactory share of population behind NATs. Further reducing the re-registration window will cause substantial increase in bandwidth consumption. See Section *SBC Dimensioning and Performance Tuning* for additional details. Note that non-compliant SIP clients may fail to honor the recommendation provided by the ABC SBC and register less often. The ABC SBC will still keep their registration bindings alive and allow incoming traffic to them. However IP and transport layer connectivity may not stay alive without the intense traffic. Contacts of such disobedient clients will show red expired status in the Registration cache window as shown in Figure *Client-side Expired Registration Contact*.

The **Minimum registrar expiration** "knob" steers how much SIP traffic passes through to the downstream registrar. Basically this parameter suggests to the downstream registrar how long a registration shall remain valid. The configured value is recommendatory only: the downstream registrar may accept it or change it in its final responses to REGISTER requests. The value in the response from the downstream registrar is used as the actual time-to-live for the cached registration binding. Registration renewals are not passed from the User Agents to the registrar until this time-to-live is about to expire. The longer this window is, the less traffic will be forwarded to the registrar. On the other hand, a client's failure to renew its contact will remain undetected by the downstream registrar and result in "hanging contacts" if the client is actually unavailable.

In the normal case when reduction of the upstream rate is desirable, the ratio of registrar-to-UA must be greater than 2.0 – otherwise the server registration window will not be long enough to capture more than one client registration. Typically the ratio is higher, 10.0 at least. The throttling action MUST be placed before any other register processing actions to take effect.

It is also important to understand the time-to-live of the cached registered contacts in detail. Briefly, the bindings stay active for the time requested by SIP registrar in response to forwarded REGISTER requests. They will be deleted if any of the following conditions occurs:

- the UAC fails to re-register its contact before the time-to-live of the contact set by the downstream registrar expires. That also means that a failure of a User Agent Client to renew its contact within the "client window" are tolerated by the ABC SBC as long as the time-to-live period is not over. A "reg-expired" event is generated. (See Section *Events (optional)*) Example of a registration cache view when only the "UAC-side" timeout expires is shown in Figure *Client-side Expired Registration Contact*.
- the UAC explicitly de-registers using procedures described in RFC3261. In this case a "reg-del" event is generated.

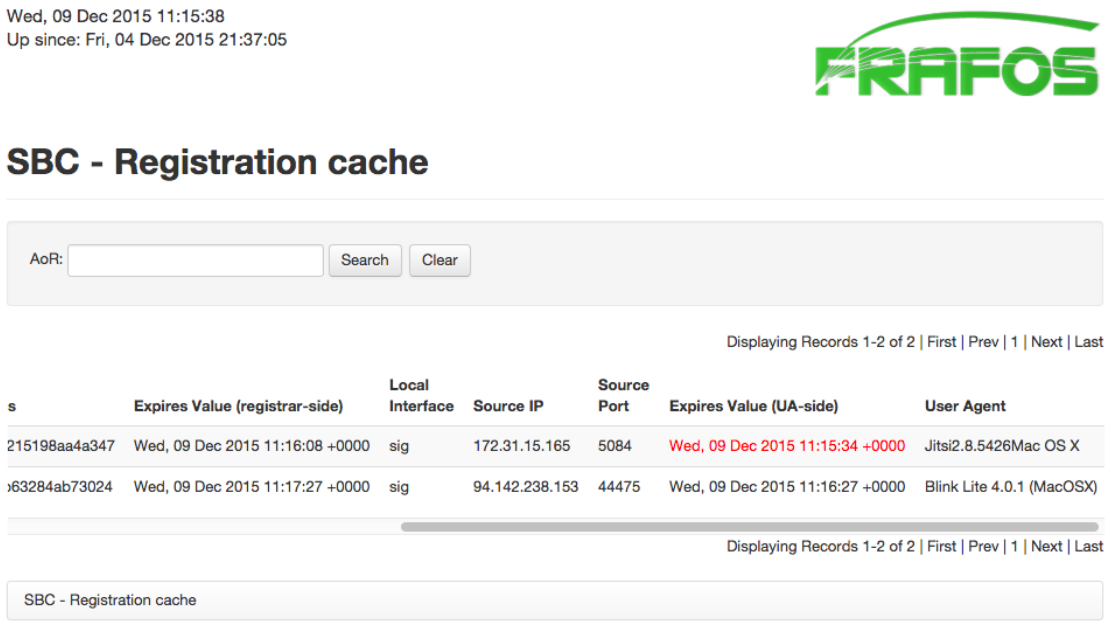


Fig. 54: Client-side Expired Registration Contact

6.11.3 Registration Caching and Handling by Example

To enable registrar caching, you must let all REGISTER requests be processed by using the actions **REGISTER throttling ** and ** Enable REGISTER caching**. The throttling action takes additional parameters: **Minimum registrar expiration** and **Maximum UA expiration**. The former specifies the "throttle" which reduces the registration traffic propagated towards a registrar. The value is normally in order of tens of minutes, we chose a whole hour in our configuration example. The other parameter, **Maximum UA expiration**, determines the traffic pace towards the SIP User Agents. The value is normally in order of minutes to keep REGISTER messages flowing and holding IP connectivity through firewalls and NATs upright. We chose an extremely aggressive value in our example, half a minute.

Figure *Registrar handling* shows a screenshot with such a 3600/30 throttling ratio configuration. The rules are part of a "public" realm serving REGISTERs coming from the public Internet.

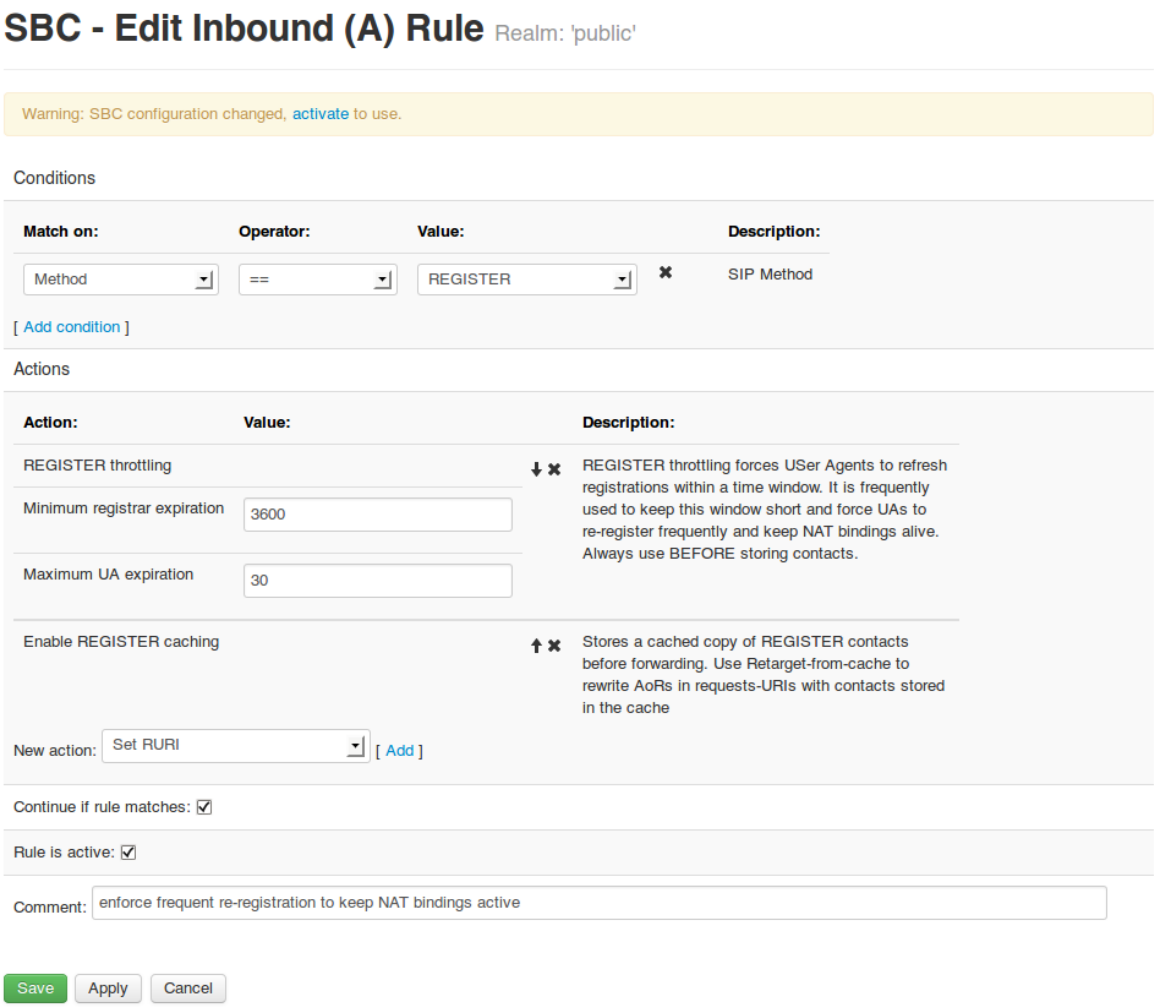


Fig. 55: Registrar handling

You also need to configure how incoming messages for the registered users will be processed. Particularly the URIs coming back in incoming requests must be recovered to the original form in the initial REGISTERs received by the ABC SBC. To do so, enable the action **Retarget R-URI from cache**, with the **enable NAT handling** option turned on for all traffic routed to the public realm. The configuration is shown in Fig. *Restoring cached contacts*.

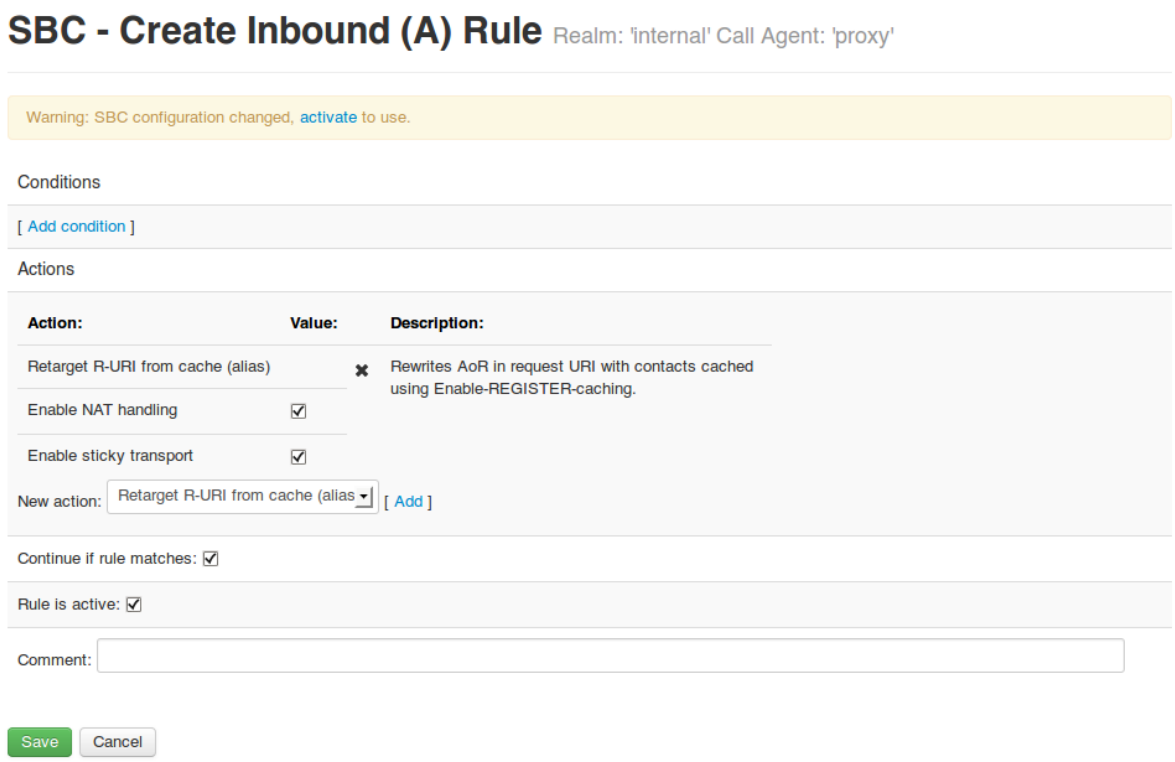


Fig. 56: Restoring cached contacts

With this configuration in place, the actual SIP call flows may appear like in the following diagram *Call Flow Registration Throttling*.

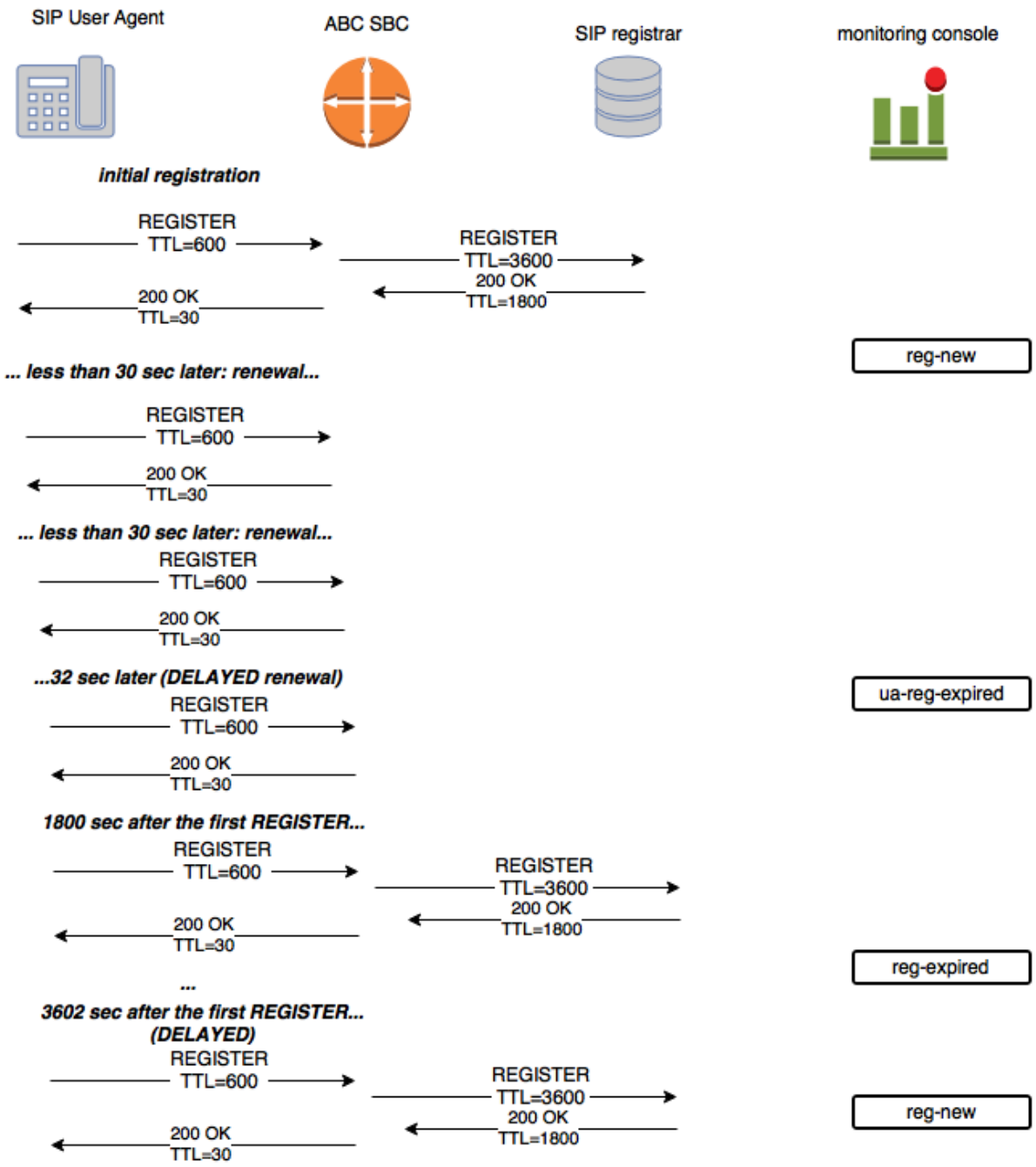


Fig. 57: Call Flow Registration Throttling

The example sequence shows the primary function of the registration throttle: increasing the traffic towards SIP User Agent (left-hand side) and reducing it towards the registrar (right-hand side). It also demonstrates how SIP equipment may differ from the expected traffic pattern and how the ABC SBC will deal with it.

The sequence begins with an initial REGISTER. The SIP telephone proposes a “time-to-live” in the message, 600 seconds in this example. (The SIP message element is really called “expires” but we found the “TTL” name more explanatory.) The ABC SBC chooses to send the traffic to downstream registrar less often, and overrides this value to a longer period of 3600 seconds. The registrar downstream finds it too high though and agrees to keep contacts for only 1800 seconds in its 200 SIP response. Now the ABC SBC knows how often it must refresh the registrations downstream: every half an hour.

In the direction towards the client, the ABC SBC compels the client to re-register more often by advertising it would only keep the registered contacts for no longer than 30 seconds.

As a result, the client keeps registering every half a minute, and the ABC SBC passes on the registration to the downstream registrar every half an hour. If the client is late with a renewal request, the address binding remains in ABC SBC registrar cache. If the client is however re-registering too late with respect to the registrar TTL, the cached registration will expire, same way like of there was no cache. The late re-registration will then create a newly registered contact.

6.11.4 Registration Agent

The ABC SBC can register itself with a third-party service using a SIP address of record. That allows it to receive incoming requests for this address subsequently. The ABC SBC does so by sending REGISTER requests periodically and authenticating them if challenged to do so. This may be for example useful, when an ABC SBC installation is configured to use the built-in conference bridge and is also supposed to serve calls from PSTN coming via a SIP-2-PSTN service.

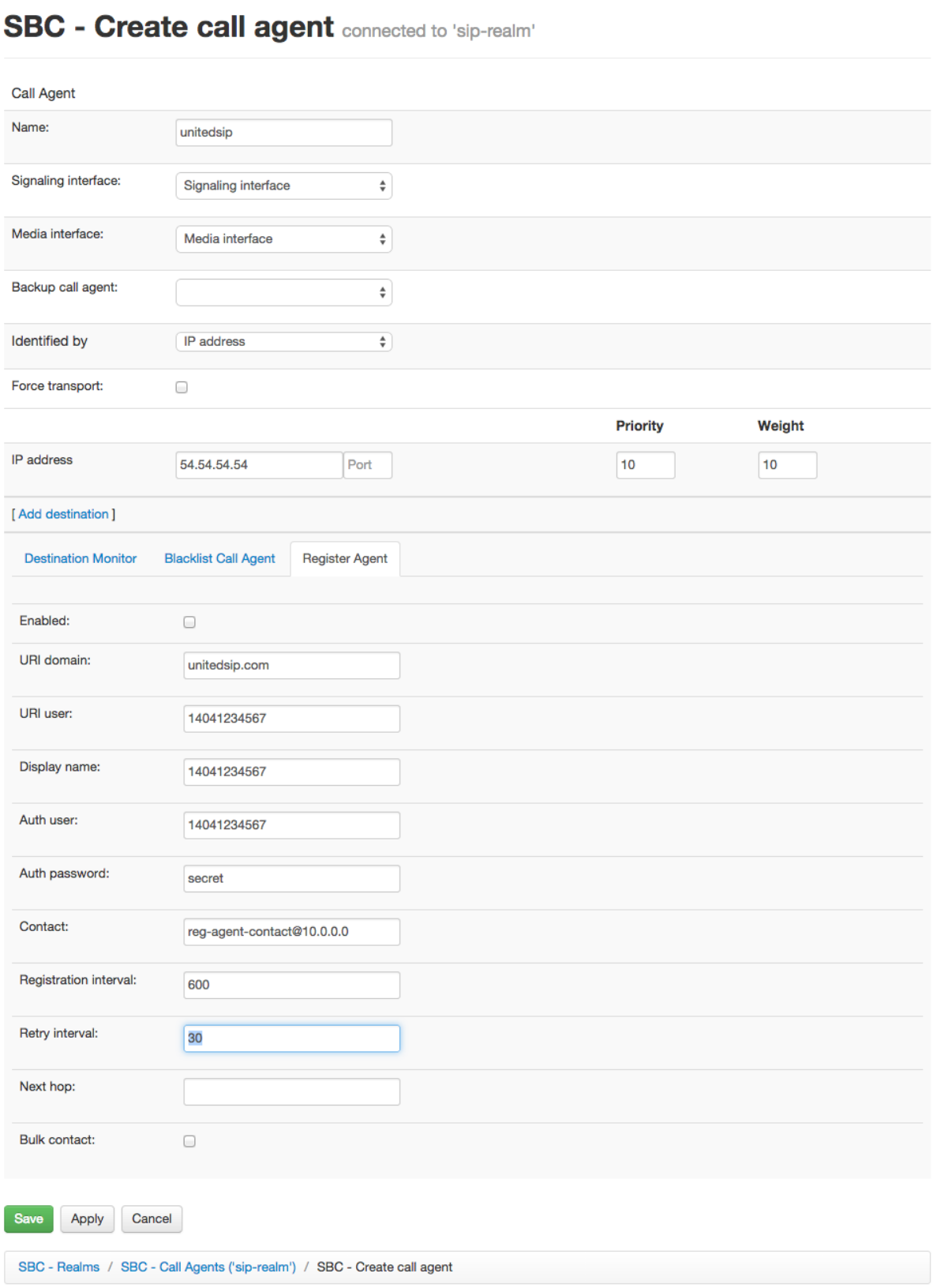


Fig. 58: Screenshot of Registration Agent Configuration

The status of registration agent can be inspected under **"Monitoring → Registration Agents"** as shown in Figure *Screenshot of Registration Agent Monitor*.

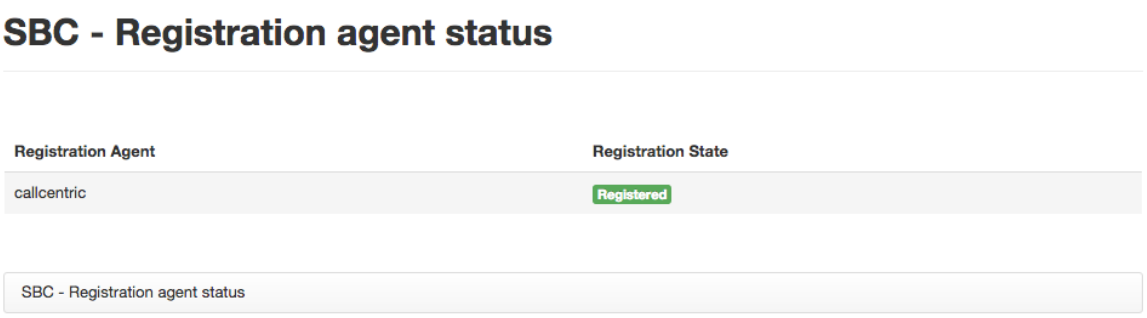


Fig. 59: Screenshot of Registration Agent Monitor

6.12 Call Data Records (CDRs)

The ABC SBC generates Call Data Records (CDRs) for every call processed by the SBC.

For conference CDRs (experimental), please refer to *New restify CDR process*.

6.12.1 CDRs Location

CDRs are generated into the directory:

```
/var/log/frafos/cdr/
```

They are generated on Source Realm basis, so every CDR is filtered to a specific file with the name „*cdr-source_realm_name.log*“. All CDRs also go into one combined file called „*cdr.log*“.

CDR output files are rotated once a day at midnight and exported to the archive directory: :

```
/var/log/frafos/cdr/export
```

The exported files are renamed to include the date – e.g. ” *cdr.log-20120701*“. The files are stored for 93 days by default and then are deleted from the disk.

The number of daily rotated files to keep and also the directory for exported files can be changed using Config / Global config, using settings under “CDRs” tab. The lowest possible number of days to keep the exported CDR files is one day.

6.12.2 CDR Format

CDRs are stored in CSV format and contain following items in given order:

- Source Realm
- Source Call Agent
- Destination Realm
- Destination Call Agent
- From user part
- From host part
- From display name
- To user part
- To host part

- To display name
- Local tag (ID for call)
- Timestamp when the call was initiated (format - 2012-05-04 02:22:01)
- Timestamp when the call was connected (format as above)
- End Timestamp of the call (format as above)
- Duration from start to end (sec.ms)
- Duration from start to connect/end (for established/failed call; sec.ms)
- Duration from connect to end (for established call; sec.ms)
- SIP R-URI
- SIP From URI
- SIP To URI

CDR example:

```
pstnprovider.com,gw1,mobile.com,uas,"alice","example.com","","bob","192.168.1.4","
↪",
"6D47CCAA-4FF10747000824C5-80299700","2012-07-02 04:28:23","2012-07-02 04:28:28",
"2012-07-02 04:28:33","10.139","4.895","5.244","bob@192.168.1.4:6000",
"alice@example.com", "bob@192.168.1.4"
```

6.12.3 Access to CDRs

CDRs can be accessed remotely using SFTP protocol. There is a default user configured with privileges to the particular location.

```
username: cdr
```

The password has to be set manually on the ABC SBC node, using the following command:

```
% passwd cdr
```

Using SFTP you can log in from your local machine to the FRAFOS ABC SBC and download the CDR output files:

```
% sftp cdr@<SBC IP address> % sftp> get -r *
```

To get only exported files (i.e. files that are not updated any more and are ready for post-processing):

```
% sftp> get export/*
```

6.12.4 Customised CDR Records

The content of CDR records can be changed using configuration file :

```
/etc/sems/cc_syslog_cdr.conf
```

Only the value of „*cdr_format*“ option (the CDR structure) can be changed. Changing any other options may cause CDR subsystem malfunction and should be done by authorised person only.

After changing the CDR configuration file, the SEMS process of the ABC SBC needs to be restarted manually.

Following items can be used in *cdr_format* option:

- $srcrlm.name - Source Realm

- $srcca.name - Source Call Agent
- $dstrlm.name - Destination Realm
- $dstca.name - Destination Call Agent
- caller_id_user - From user part
- caller_id_host - From host part
- caller_id_name - From display name
- callee_id_user - To user part
- callee_id_host - To host part
- callee_id_name - To display name
- $ltag - Local tag (internal call identifier)
- $start_tm - Timestamp when the call was initiated (format - 2012-05-04 02:22:01)
- $connect_tm - Timestamp when the call was connected
- $end_tm - End Timestamp of the call
- $duration - Duration from start to end (sec.ms)
- $setup_duration - Duration from start to connect/end (for established/failed call; sec.ms)
- $bill_duration - Duration from connect to end (for established call; sec.ms)
- sip_req_uri - SIP R-URI
- sip_from_uri - SIP From URI
- sip_to_uri - SIP To URI
- disposition - Result of call establishment. Possible values: answered,

failed, canceled.

- invite_code - Result code of final reply to initial INVITE (not set for

canceled calls).

- invite_reason - Reason phrase of final reply to initial INVITE (not set for

canceled calls).

- hangup_cause - Reason for the call termination, set for established calls

only. Possible values: BYE, reply, no ACK, RTP timeout, session timeout, error, other.

- hangup_initiator - Reason for the call termination. It is set for answered

calls only where hangup was caused by request (BYE) or in case of call was terminated because of local error. Possible values: caller, callee, local

- ucid - Unique Call Identifier. Can be used to map CDRs for transferred calls together. The value is common for all CDRs generated for calls that are results of unattended call transfer from one original call done by the ABC SBC.
- call variable - $gui.<call variable name> - Allows to write user specified

call variable into CDR. For example $gui.experimental_variable will write the value of experimental_variable into CDR. These outputs are located under "/var/log/frafos/cdr/cdr.log". The call variable can be set using "**Set Call Variable**" action, see *Binding Rules together with Call Variables* for more details on using call variables.

Please make sure "List of call variables added into events:" section is filled by requested call variable to see in CDR. Under "Call Events" table, ADVANCED button should be clicked to be able to see call variable on Monitor as requested.

Important: CDRs are written using syslog and split into per-realm files according to first item that is expected to be the source Realm. If you change the CDR format the way the first item is not a "realm" name, the files will be named according to the values in this column and won't represent per-realm data any more.

6.13 Advanced Use Cases with Provisioned Data

The ABC SBC can be integrated with external or internal sources of data and logic. This allows to complement its rigid rules-based logic with richer and more complex applications provisioned by the administrator.

The following methods are available:

- **Generic RESTful queries** to an external server using the **Read call variables over REST** action. Using this interface allows to drive the ABC SBC behaviour using in-house developed business logic located in external web programming environments. see Section *RESTful Interface* for more details.
- **Provisioned tables**. Solving some problems with tabular nature, like Least-Cost-Routing, Blacklisting, Dial Plan Normalization or SIP Connect Bulk Registration is much easier if tables are provisioned separately from the rules. For this reason the ABC SBC supports on-board provider-provisioned databases. See Section *Provisioned Tables* for more details.
- ENUM queries using the **Enum query** action , as described in the section *ENUM Queries*. This method allows for a number-to-URI translation using the DNS-based ENUM queries.

6.13.1 RESTful Interface

The RESTful interface embedded in the ABC SBC allows high programmability of the SIP Sesion Border Controller.

The interface addresses an important dilemma for operators: how to introduce new scenarios, while preserving the existing ones intact. Hardwired product logic compels the operators to request code changes from vendors. Change requests result in unavoidably tedious process, weeks or months of negotiation, changes of changes and delays regardless how small and reasonable the changes are. Therefore ABC SBC comes with the possibility to implement business logic outside the product in an operator-controlled environment.

This capability follows a general trend in which the business logic is concentrated in a single place that defines behavior of relatively "dumb" network elements. The business logic defines security policies (who can call whom), marketing campaigns (at what price), and network behavior (how to route the calls). Placing this logic in a web server relieves operators from inadequate vendor dependencies and allows PHP, Perl, and virtually any web programmer to implement new SIP scenarios in a well understood programming environment.

The operation of a RESTful application is simple and consists of three steps characteristic for any computer program. The steps are depicted in Figure *RESTFul Call Flow*: The ABC SBC receives an incoming INVITE on input in the first step A, processes it in step B, and generates a correctly processed INVITE on the output in step C. The processing in step B is split in three phases: - B.1: RestFul query is formulated that contains all pieces of SIP information needed to execute the web-based logic. The information is passed in form of URI parameters to the RESTful server. - B.2: The RESTful server performs the application logic. - B.3: Eventually the RESTful server sends an answer back to the ABC SBC. The answer contains an array of variables that represent an advice to the ABC SBC how to handle the message in the final step C.

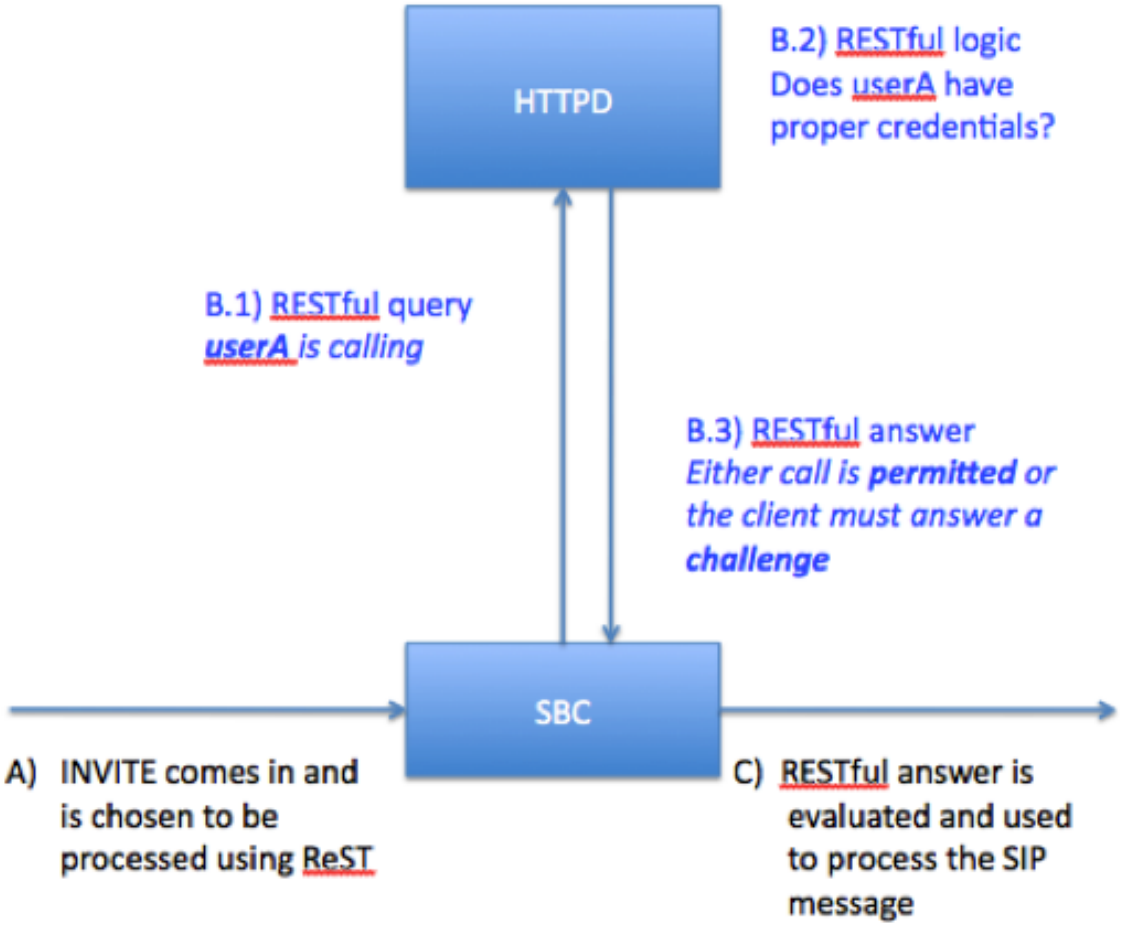


Fig. 60: RESTFul Call Flow

RESTful Interface using Digest Authentication Example

In this example we show how to outsource digest authentication to the external RESTful server. This relieves the ABC SBC of implementing a user:password database. It is even designed in a way that leaves the ABC SBC unaware of the cryptographic authentication protocol: all it does is it shuffles header-fields back and forth.

Background: digest authentication in SIP works by conveying shared password from client to server in a hashed form. If both client and server hash the password and obtain the same result, identity of client is proven without sending the password in clear-text.

The whole process follows the steps outlined above. It begins when a SIP request comes in. (only fragment shown):

```
INVITE sip:music@abcsbc.com SIP/2.0.
 Via: SIP/2.0/tcp 192.168.178.22:54251
 From: "foo" <sip:foo@abcsbc.com>;tag=0omQsGfHsCtP8-5k2.t4uJI3ekc66bGZ.
 To: <sip:music@abcsbc.com>.
 CSeq: 14693 INVITE.
 Proxy-Authorization: Digest username="foo", realm="abcsbc.com",
   nonce="UP6CiVD+gk0Uyu4WHAv+48ypPC2vjH+6", uri="sip:music@abcsbc.com",
   response="560ad1cc8777efa6a6cc1857795ec155".
```

The INVITE request signals a call from user with address sip:foo@abcsbc.com to address sip:music@abcsbc.com The ABC SBC checks the request against its rules and initates the RESTful logic. (see Figure *Rule for Evoking a RESTful query*).

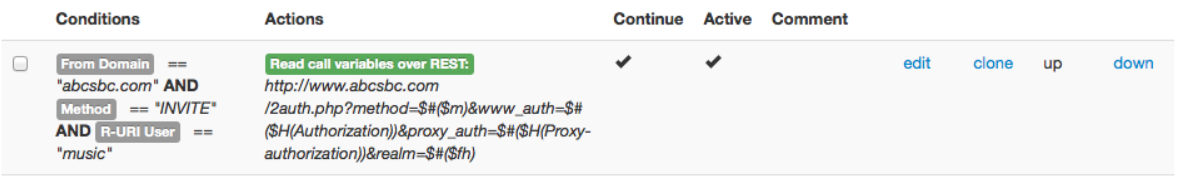


Fig. 61: Rule for Evoking a RESTful query

The rule in ABC SBC's configuration matches by From domain, method and request URI, and therefore processing is passed to the action "Read Call variables over REST". ABC SBC is configured to pass several headerfields as URI parameters to the RESTful application. Particularly, the user information relevant to authentication are passed: Authorization and Proxy-authorization header fields ($H(Proxy-authorization)), request method ($m) and realm. The domain in From URI ($fh) is used as realm – this way you can build up a multidomain hosted service, which will work same for any domain without change.

Now the SBC has received a call, chosen to process it using a web server, the REST can begin by sending an HTTP query. Let's see how the query looks on wire. It simply conveys the values chosen in SBC configuration as URI parameters. Symbols contained in the values are substituted using the escape code %:

```
GET /2auth.php?method=INVITE&www_auth=&proxy_auth=Digest+username%3d%22foo%22%2c
+realm%3d%22abcsbc.com%22%2c+nonce%3d%22685f3174-6aaf-4337-a9f4-4cf4d1f150ab%22%2c
+uri%3d%22sip%3amusic%40abcsbc.com%22%2c
+response%3d%225b3cc376e815c949bbc084c747a3a55f%22&realm=abcsbc.com HTTP/1.1.
User-Agent: REST-in-peace/0.1.
Host: www.abcsbc.com
```

When the web server receives this query, it starts an application. In our example we have chosen to build it using PHP, it could be done same well using Perl, Java, python or any other popular web programming language. In its own way the interpreter passed the URI parameters to application's variable and processing begins.

The following PHP code shows key steps during computation. Not all are shown. For a given user, it calculates her hashed password stored in database and checks it against one coming in the request. If they are equal, it answers with a 200 answer suggestion, otherwise it advises the SIP server to reauthenticate the user:

```
// simulation of a database query ... ask username and password
$users = array('foo' => '12', 'guest' => 'guest');
// prepare challenge for the case credentials are invalid
// or missing
$challenge='Digest realm="'.$realm.'",nonce="'.new_nonce().'"'

// no credentials supplied? Request some!
if (empty($proxy_auth)) {
  print_answer("407", "Authenticate",
         "Proxy-Authenticate: ".$challenge,$cmt);
}
// parse the credentials
$data=parse_hf($proxy_auth);
// calculate expected answer
$expected_response=calculate_answer($data);
if ($data['response'] != $expected_response) {
  print_answer("407", "authenticate", "Proxy-authenticate",
                $challenge);
  return;
};
// otherwise proceed with OK
print_answer("200", "ok");
```

Now we have an answer: either a positive 200, or a negative 407 with authentication challenge to be passed to the SIP client through the ABC SBC. If you observe the wire you will see the following HTTP answer:

```
HTTP/1.1 200 OK.
Date: Tue, 22 Jan 2013 11:58:47 GMT.
Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch6 mod_ssl/2.2.3 OpenSSL/0.9.8c.
X-Powered-By: PHP/4.4.4-8+etch6.
Content-Length: 214.
Content-Type: text/html.
.
code=407.
phrase=authenticate.
headers=Proxy-Authenticate: Digest realm="abcsbc.com",
   nonce="685f3174-6aaf-4337-a9f4-4cf4d1f150ab"\r\n.
```

What you see here in clear-text is, that the programmer has stored the processing results into several variables that are passed back to the ABC SBC rule-base.

We are in the final stage now – the web application has returned processing results back to ABC SBC, the SBC will evaluate the parameters in its rules and use them in further SIP processing. Particularly, the rule in Figure *Rule for processing result of RESTful query* says, if processing ended up with variable code not being equal to 200, the call will be refused. The negative answer will include parameters determined in the HTTP answer.

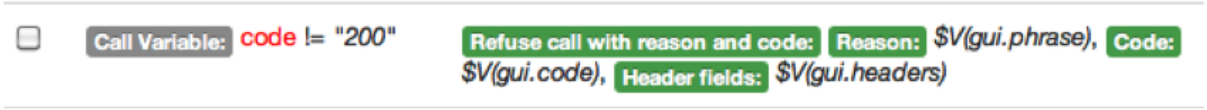


Fig. 62: Rule for processing result of RESTful query

Here is the final outcome of our effort: SIP answer calculated in the RESTful server and challenging SIP client to submit proper credentials:

```
SIP/2.0 407 Authenticate.
Via: SIP/2.0/tcp 192.168.178.22:54251;received=83.208.91.146
From: <sip:foo@abcsbc.com>;tag=0omQsGfHsCtP8-5k2.t4uJI3ekc66bGZ.
To: <sip:music@abcsbc.com>;tag=50123210ee9d5f0f8df05cf1f196cfeb-c5a6.
CSeq: 14692 INVITE.
Proxy-Authenticate: Digest realm="abcsbc.com", nonce=
↪"UP6CiVD+gk0Uyu4WHAv+48ypPC2vjH+6"
```

6.13.2 Provisioned Tables

The ABC SBC rules can refer to an internal database maintained separately from the rules logic. This greatly simplifies use-cases which would have to be implemented using a large numbers of almost identical rules otherwise. The typical use-cases include tests if a URI is on a blacklist or list of monitored users, static SIP registrations, Least Cost Routing tables, definition of dialing plan normalization and more.

The tables are physically located on the ABC SBC machine for highest performance, can be provisioned using the web interface and can include any number of administrator-chosen attributes in addition to the lookup key. There is also a possibility to provision the data remotely via RPC.

There are two types of tables:

- **data** - general purpose data tables can be queried to fetch specific data associated with a key. The structure of such tables can be freely defined by the administrator, thus allowing great flexibility. The data tables are used from A-rules and C-rules.
- **routing** - specialized routing tables have a list of mandatory attributes that define routing behaviour and are always present. Additional attributes may be added. The routing tables can only be used from B-rules.

Using the tables is as simple as creating a table with the desired structure, filling it with data, looking up a result in the table from A,B or C-rules by a selected value, and processing the found data entry. The data entry is returned

to the script processing as variables bearing the names of the table columns. The whole process is described in the following subsections in detail.

Please be aware that restful provisioned tables queries have some limitation compared to the one available via GUI. The rest matching cannot emulate a case insensitive match as the data are stored in a redis database, while it's a sql for the later one.

Configuring Tables

The process of setting up a new provisioned table consists of the following steps:

- **Analysis of the problem to be solved**. You need to specify what data you are going to lookup in a table by what key and how you are going to use the resulting table record.
- **Definition of the table structure**. This is started from the Web menu under *Provisioned Tables* → *configure* → *Insert new*. There you must identify:
  - key lookup operator which is one of *equal*, *range*, and *prefix*. The operator defines the method by which a key is looked up in a table. It is not possible to lookup in the same table using some other method. If *range* is chosen, the resulting table will include two key columns for begin and end of a range. If *prefix* is used and overlapping choices are found, the longest match is selected.
  - table keys and their types. For *range* and *prefix* lookup operator just one key could be defined. For *equal* operator multiple keys could be defined. The key type is one of *uri*, *number*, *call-agent*, *string*. The type is used for syntactical checking when the actual data is entered later. Even more importantly it is used to determine how the lookup operator is used. Particularly, prefix lookup is sensible for string types as it discriminates between "0" and "00". For numerical types, these two values would be the same.
  - and type of table *data* or *route*, as explained above.
  - There is also Group-by, which can be *none* or *string*. The option *string* allows to add an informational tag to each entry so that tables can be viewed by groups.
  - Optionally, any number of additional table named columns can be added, whose type must be chosen from *uri*, *number*, *call-agent*, *string*. When the "save" button is pressed, the table structure is created and becomes instantly ready for filling with data.
- **Filling tables with data**. This is started from the Web menu under *Provisioned Tables* → *(table name)*. On the webpage that opens, the link *Insert new rules* opens a dialog for inserting a new data entry. When editing the new entry is complete, pressing the "Save" button will store it. A Version 4 UUID (**RFC 4122**) is automatically added to every entry for sake of internal data maintenance.
- **Completing data entry**. To make the ABC SBC understand that the newly created records can be used, the button *Activate Changes* must be pressed. This allows editing the tables without interfering with the table as currently being used by the ABC SBC. *Activate Changes* activates the current version of the table for use by the ABC SBC, and creates a new table version which is used for further provisioning.
- **Introducing the table lookup in the rules**. This requires adding the action *Read Call Variables*. The action takes the table name as the first parameter and lookup value as the second. The lookup value is typically formed using substitution expressions (see Section *Using Replacements in Rules* for a reference.)

Provisioned Table Example: Static Registration

We will start with a relatively simple example: static registration. Similarly to how SIP devices use SIP REGISTER messages to create a temporary binding between SIP Address of Record and actual Contact URI, administrators can provision a similar association manually and permanently. That means that a table is provisioned and used the following way:

- The table is keyed by an Address of Record URI and includes the next-hop URI as attribute.
- When a SIP INVITE comes in, its request URI is checked against the table and if the next-hop is found, replaced with it.

Note that this structure can be used to implement static call-forwarding.

Screenshots of the resulting table structure, table content and rules using the table are shown in the Figures *Structure of Static Registrations*, *Static Registration Records*, and *Static Registration Rules* respectively.

You can make several observations about the table lookup rule:

- The rule conditions make sure the lookup is only executed for authenticated INVITEs, which helps to eliminate unnecessary database queries - there is no point in making the table query for other than INVITE requests. If authentication is requested, the lookup for the first unauthenticated request would be also useless.
- The table entry is looked up by the replacement expression "$r." which stands for the current request URI. The result is returned in variable next_hop, as defined in the table column name.
- If no result is found, the table lookup placed in the rule's condition will return FALSE and no action will not be performed.
- If a result is found, we apply two actions: change request-URI and add a header-field for troubleshooting purposes. Its presence in the outgoing INVITE shows a lookup was performed, the request-URI which was used as key and the returned value.

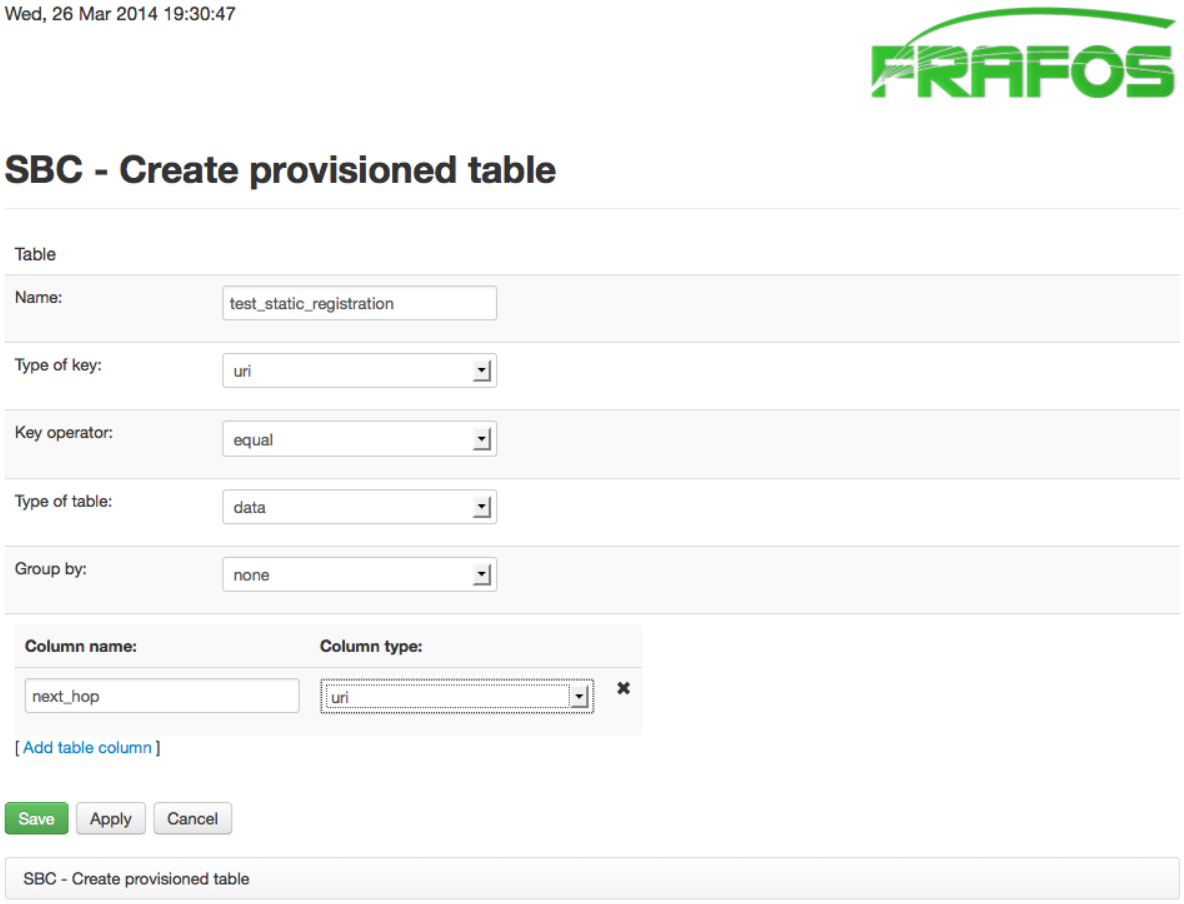


Fig. 63: Structure of Static Registrations

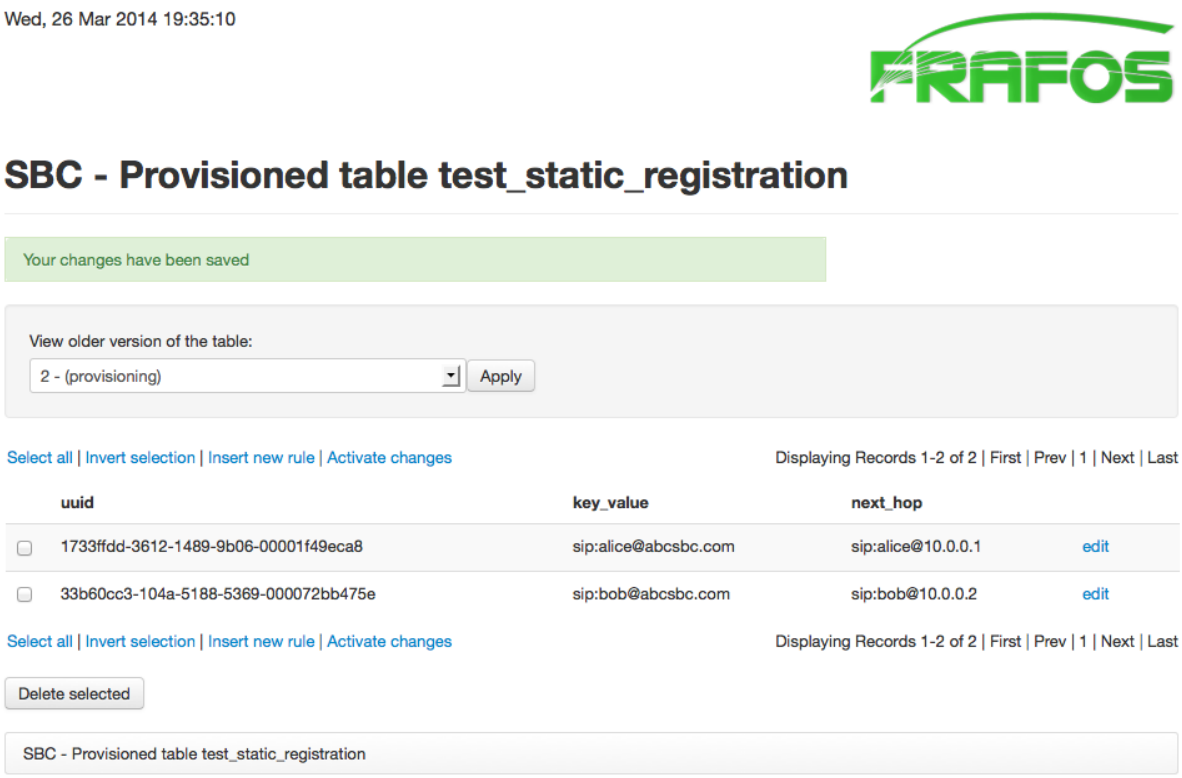


Fig. 64: Static Registration Records

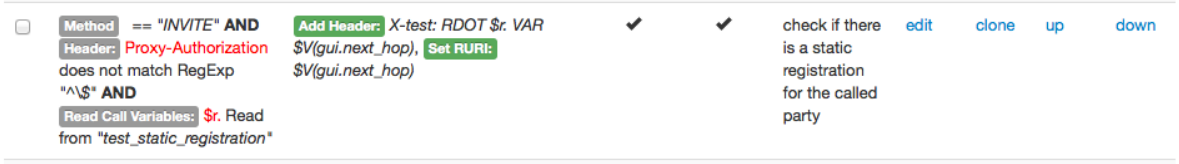


Fig. 65: Static Registration Rules

Provisioned Table Example: URI Blacklist

Simple tables can have a great use. In this example we test presence of a SIP element value on a list. That means that the table only includes keys, with which no additional values are associated. We then look up the elements in the keys. That can be used to implement scenarios involving all kind of discrimination like:

- Call recording: is a call coming from a user whose calls shall be recorded?
- Domain discrimination: is a call being routed to a listed domain for which some header fields must be removed or appended?
- URI Blacklisting: is the caller blacklisted?

The following screenshots show the configuration of the URI-blacklisting example: Figure *URI Blacklist Structure*, Figure *URI Blacklist Content*, and Figure *Blacklisting Rule*. The rule is simple: If the SIP URI in the From header-field matches a URI in the blacklist table, the request is declined using a 403 response. Note that the lookup key is concatenated using “sip:” and “$fu”, because the replacement expression $fu does not include a protocol discriminator.

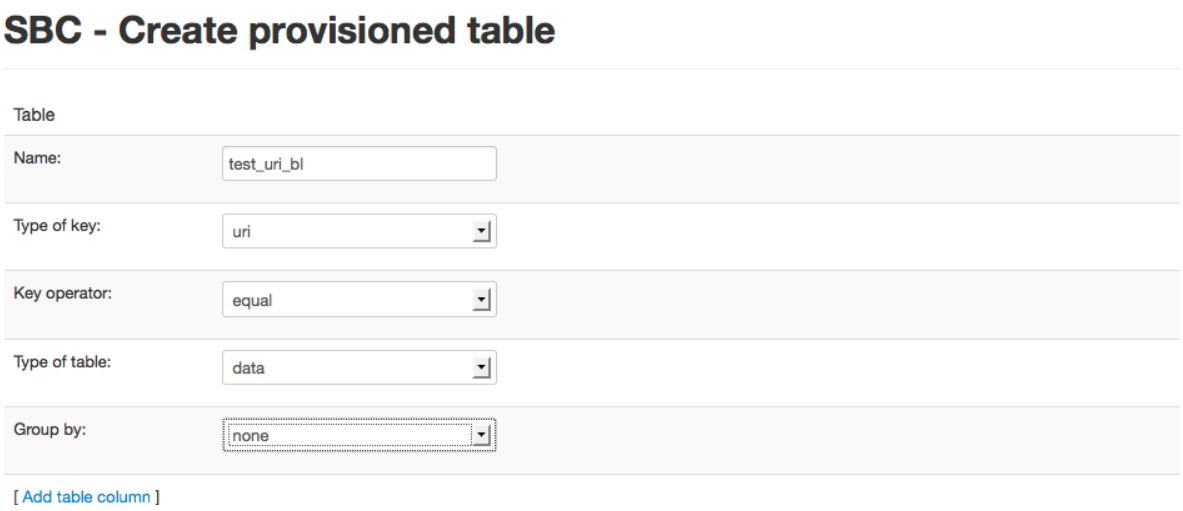


Fig. 66: URI Blacklist Structure

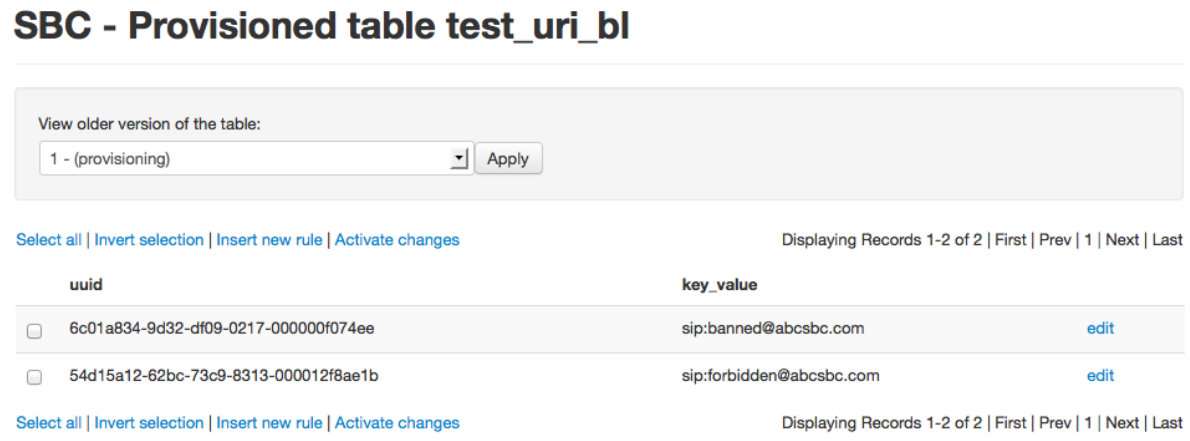


Fig. 67: URI Blacklist Content

Fig. 68: Blacklisting Rule

Table Example: Dialing Plan Normalization and Least-Cost-Routing

This is a two-in-one example showing two tables that are usually cascaded behind each other: normalization of a PBX dialing plan and least-cost routing.

Telephone numbers as used within a PBX can have different forms, following local national conventions and enterprise policies. For example, a typical user of a PBX in Munich dials with three leading zeros followed by international and area code to reach an international destination, two zeros followed by area code to reach destinations within Germany, one zero to reach destinations within Munich metropolitan area, and phone numbers without leading zeros to reach other PBX users. Using this dialing convention is convenient, the number length only grows with distance. However, these numbers loose significance if one tried to use them globally say to reach an international PSTN gateway. Therefore it is useful to normalize them in the E.164 format by stripping leading digits and introducing an appropriate prefix.

The following table shows examples of telephone numbers and how they are normalized for calls from a Munich PBX:

local number number	E.164 equivalent	digits to be stripped	prefix to be introduced
000140433345678 (US destination)	+1-404-333-45678	3	+
003034567000 (German number)	+49-30-3456-7000	2	+49
078781234 (Munich number)	+49-89-7878-1234	1	+4989

The following screenshots show the configuration of dialing plan normalization: Figure *Dialplan Structure*, Figure *Dialplan Content*, and Figure *Dialplan Rules*.

Note that the key is defined as string to make sure that prefix "00" in request URI does not match all of "0", "00" and "000" as it would if the data type would be numerical.

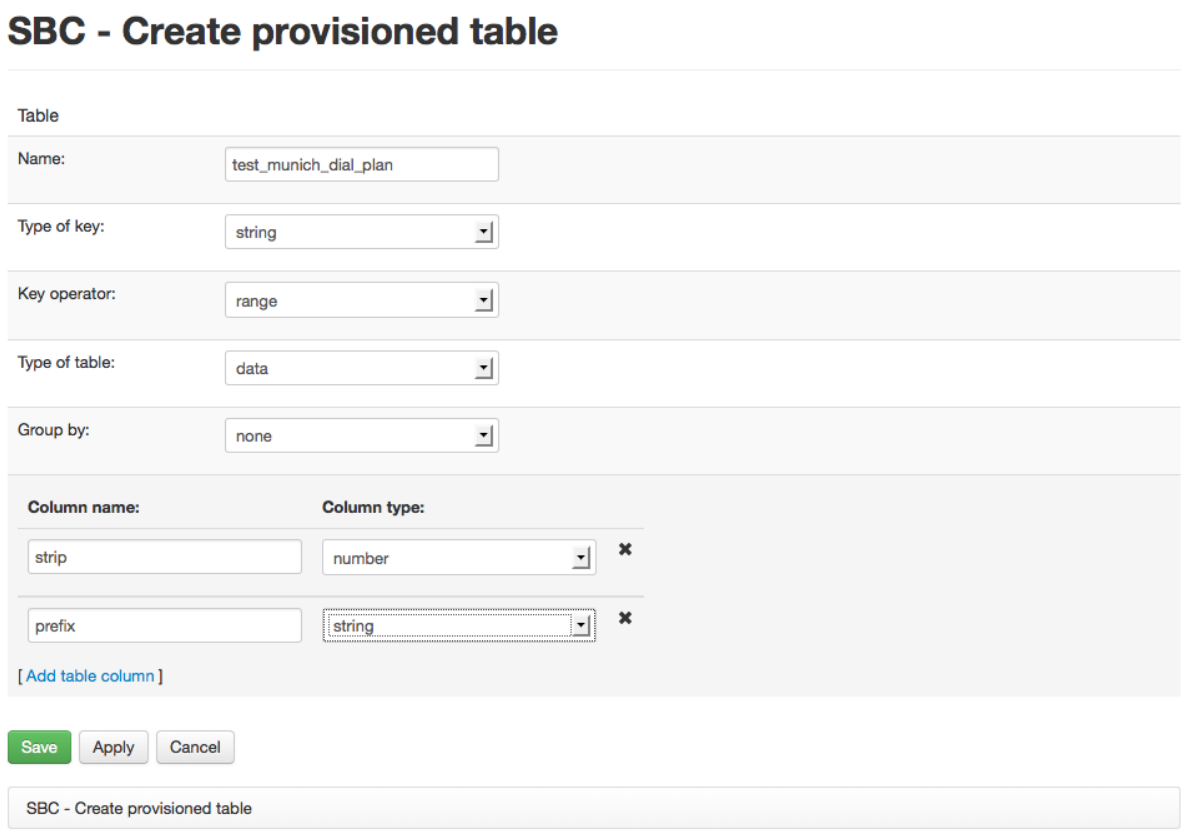


Fig. 69: Dialplan Structure

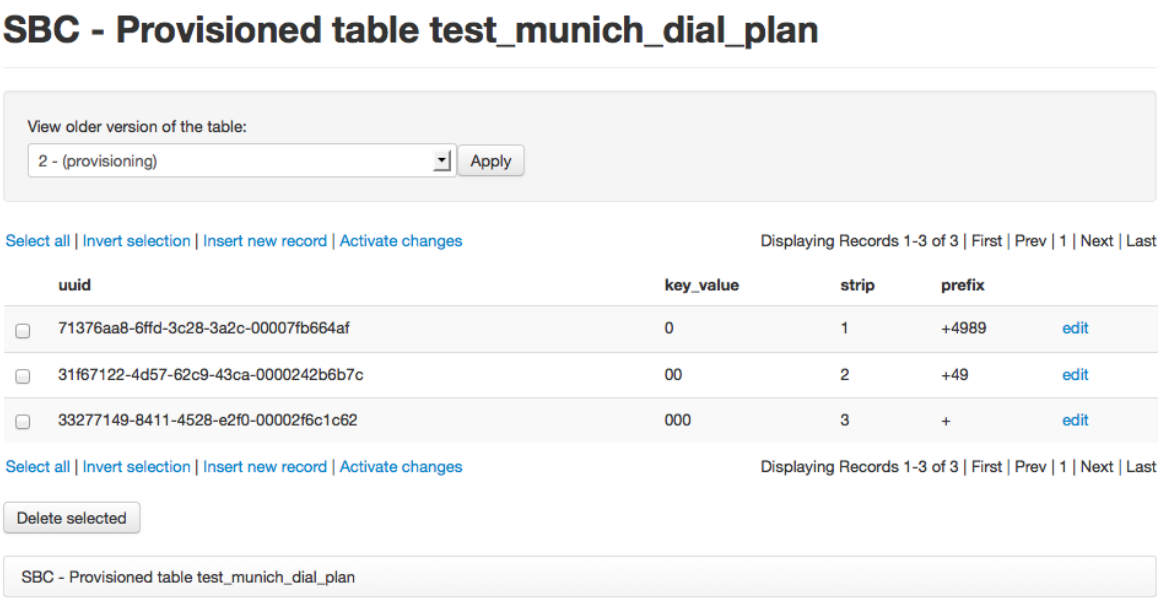


Fig. 70: Dialplan Content

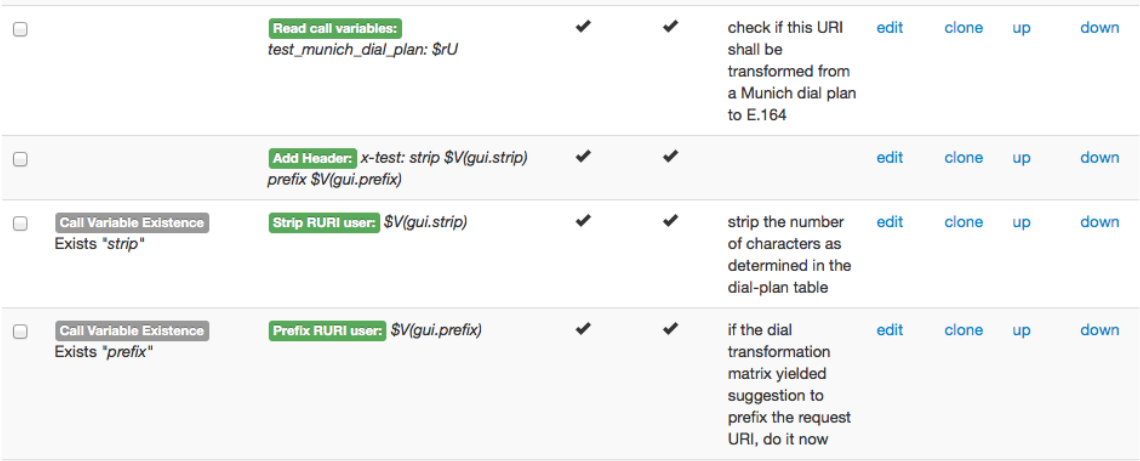


Fig. 71: Dialplan Rules

Once the numbers are normalized in the E.164 form, it is also easy to check the destination against a least-cost routing table to find the most economic PSTN gateway. The table may have the following content: prefix that is used to march phone numbers, and DNS name of a gateway chosen to serve the matched destination. Longest match applies which means that the shortest-match is taking lowest precedence and is used as "default route".

prefix	destination	comment
+1	us-gateways.com	US destinations
+43	austrian-united.com	German destinations
+	cheap-pstn.net	Everything else

The provisioning process is shown in the following three Figures: *Creating an LCR Table*, *Creating LCR Table Entries*, and *Calling the Routing table from routing rules*.

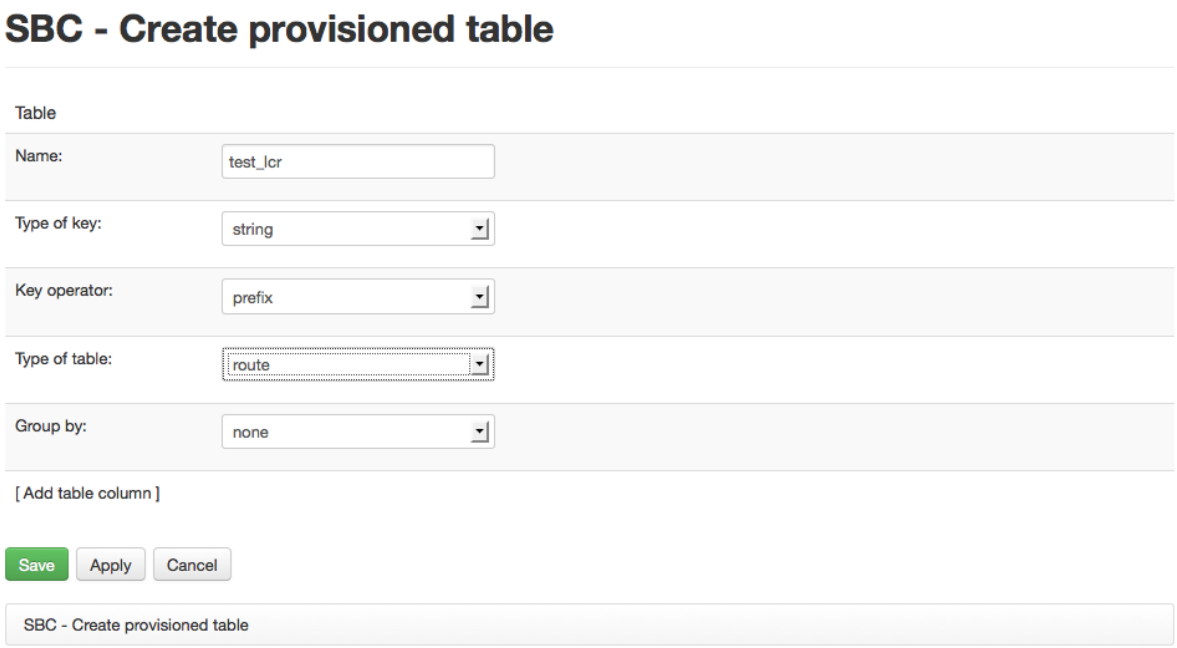


Fig. 72: Creating an LCR Table

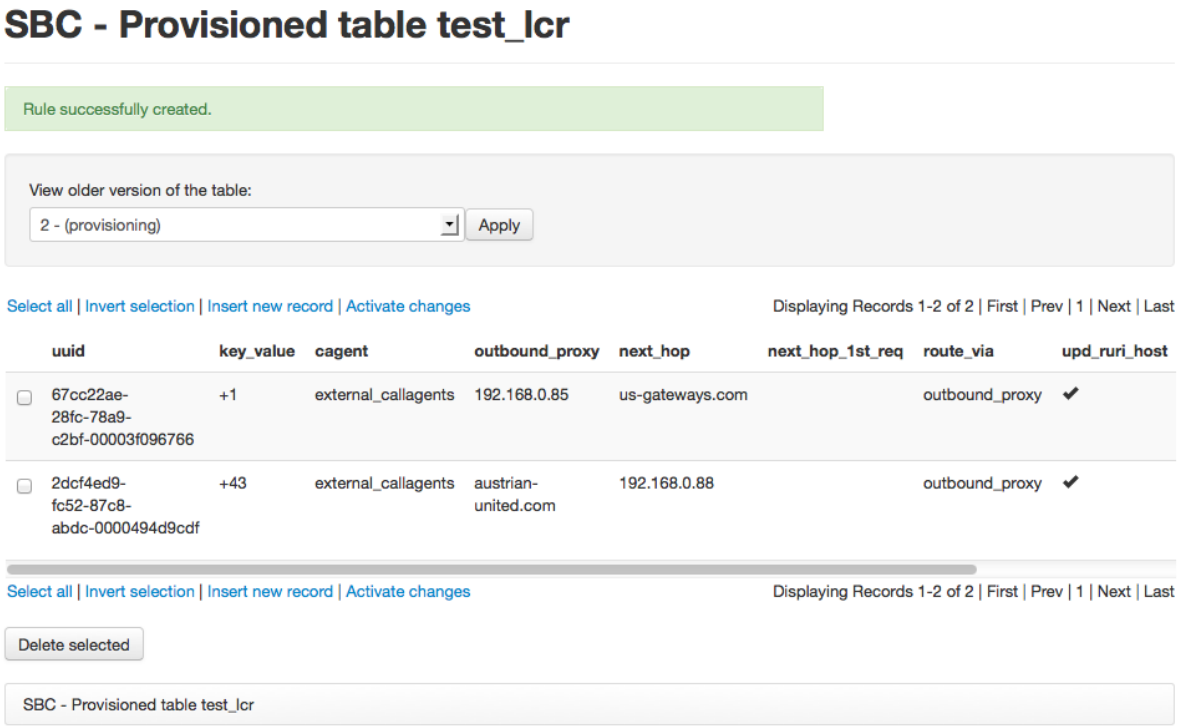


Fig. 73: Creating LCR Table Entries

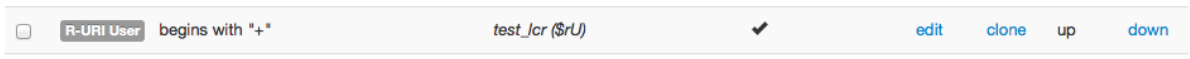


Fig. 74: Calling the Routing table from routing rules

Table Example: Bulk Registration

Thanks to an extensions of the SIP standard, it is now possible for a PBX to have one digest identity under which it can serve a whole range of telephone numbers. The extension emerged out of SIP Forum's effort to create a profile for PBX interoperability, that became known as "SIP Connect" and standardized as **RFC 6140**.

The ABC SBC supports these scenarios and it even makes the deployment scenario much simpler than contemplated in the RFC. It allows an arbitrary SIP client, PBX, softphone or any other SIP device to authenticate under a single URI and receive calls for a whole range of telephone numbers. It works "as is" without requiring any of the "bnc", "gin" or GRUU extensions.

The following example call flow assumes a network topology in which the ABC SBC guards an internal network, in which a combined proxy/registrar is located. The administrator has provisioned the telephone number range 7200-7400 to be server be PBX reachable under the URI sip:pbx@abcsbc.com. Note that a similar scenario could also be implemented using the ABC SBC's built-in registrar.

The call flow starts with a SIP registration using digest authentication (1)-(4). When an INVITE comes in (5), the telephone number in the Request URI is translated to that of the PBX (6). This allows the proxy/registrar behind the SBC to perform user-location lookup and forward to the PBX through the ABC SBC (7). The SBC then, as usual, retrieves the original URI, whose username is fixed eventually to be the target telephone number (8):

```
                                        proxy/
Internet         |SBC|                registrar              |SBC|                SIP␣
↪PBX
|                  |                      |                    |                      ␣
↪|
|                  |                      |(2) REGISTER        |(1) REGISTER          ␣
↪|
|                  |                      |To: pbx@abcsbc.com  |To: pbx@abcsbc.com    ␣
↪|
|                  |                      |m:<sip:a47b6@abc>   |Contact:<pbx@10.0.0.1>
↪|
|                  |                      |<------------------|<---------------------
↪|
|                  |                      |                    |                      ␣
↪|
|                  |                      |(3) 200 OK          |(4) 200 OK            ␣
↪|
|                  |                      |------------------>|--------------------->
↪|
|(5) INVITE        |(6) INVITE            |                    |                      ␣
↪|
|sip:7271@any.com  |sip:pbx@abcsbc.com    |(7) INVITE          |                      ␣
↪|
|---------------->|x-pbx-user: 7271      |sip:a47b6@abc       |                      ␣
↪|
|                  |-------------------->|x-pbx-user: 7271    |(8) INVITE            ␣
↪|
|                  |                      |------------------>|sip:7271@10.0.0.1     ␣
↪|
|                  |                      |                    |--------------------->
↪|
```

To orchestrate this call-flow, the following configuration steps must be taken:

- A number range must be defined and assigned to the URI the PBX owns. This is done using the table-provisioning feature. The screenshots showing this process are in the Figure *Definition of the Number Range Association* and Figure *Assignment of a Number Range to a URI*.
- In a rule, incoming INVITEs (5) must be tested against the available ranges. If such a range is found, the request URI must be translated to that owned by the PBX. At the same time the telephone number must be preserved in a request-URI parameter and/or proprietary header-field (x-pbx-user here), whichever the registrar behind the SBC can better deal with. (6) The configuration is shown in Figure *Assignment of a*

Number Range to a URI.

- Before the INVITE is eventually sent to the PBX, it must include the destination telephone number in the request URI. This is done in a rule that retrieves the phone number from the request URI parameter or header-field, in which it was stored in the previous step. The configuration is shown in Figure *Retrieving the Telephone number back in request URI*.

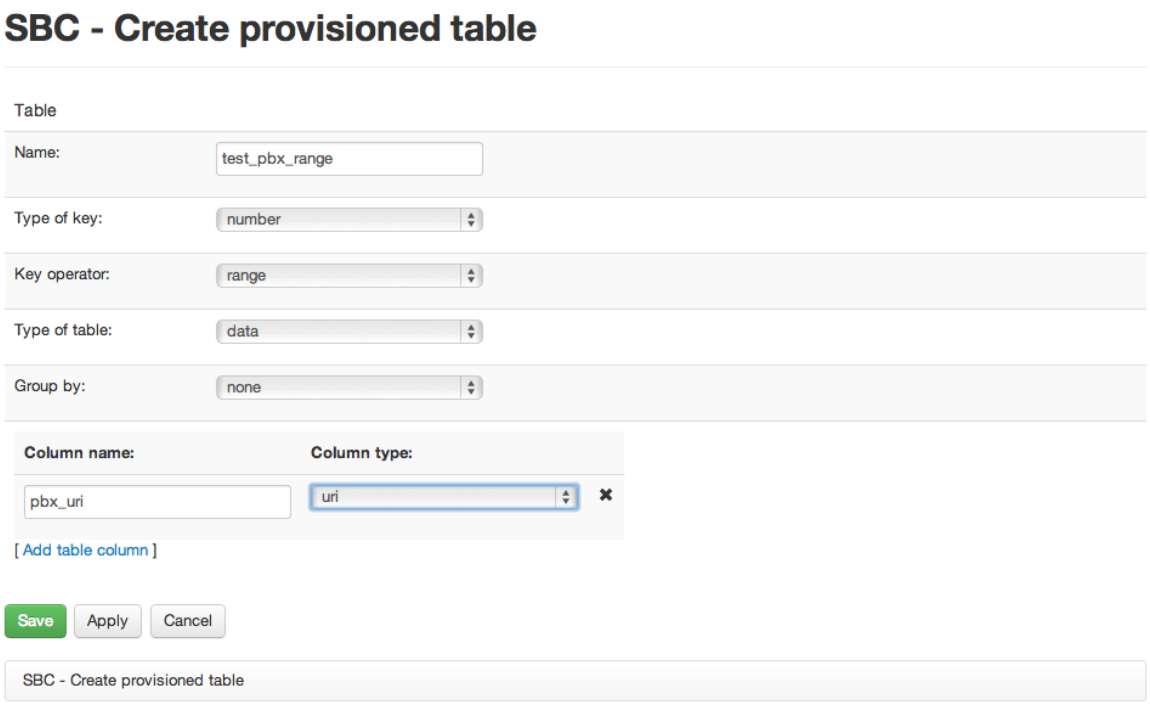


Fig. 75: Definition of the Number Range Association

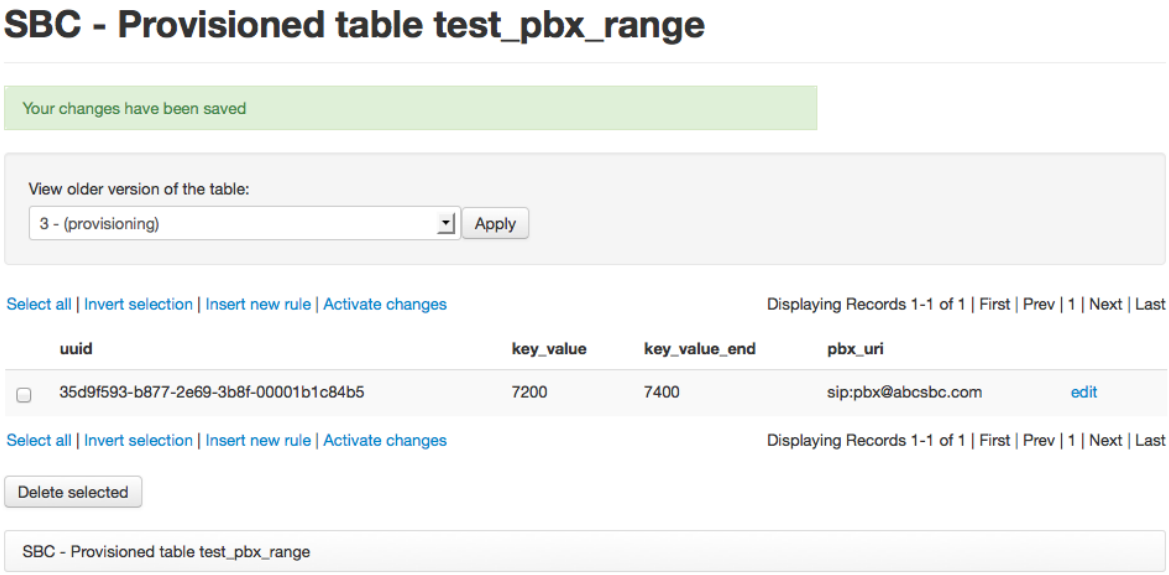


Fig. 76: Assignment of a Number Range to a URI

Fig. 77: Placing PBX's Address in request URI and Storing the Original Telephone Number

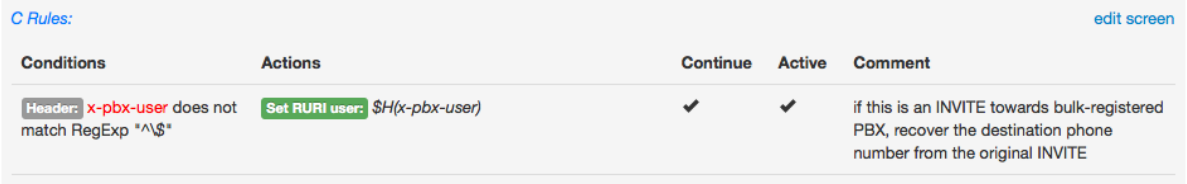


Fig. 78: Retrieving the Telephone number back in request URI

Provisioning Tables Using RPC

In the case that the ABC SBC administrator already has a table available, it will be easier to transfer it automatically to the ABC SBC as opposed to typing it in the web-interface. This can be accomplished using the ABC SBC's XML-RPC data provisioning interface.

Check following sections for more information: *Reference XML-RPC functions* and its *Provisioned Tables* subsection.

6.13.3 ENUM Queries

ENUM (**RFC 3761**) is a DNS-based phone number database that translates telephone numbers into URIs. For example, the telephone number +1-405-456-1234 can be translated to sip:mrs.somone@abcsbc.com. This is often used to find SIP address for a VoIP user when she receives a call from the PSTN under her telephone number.

An ENUM query can be run using the **Enum query** action. This action queries the default DNS resolvers configured in the SBC host with an Enum query, and sets the request URI to the result.

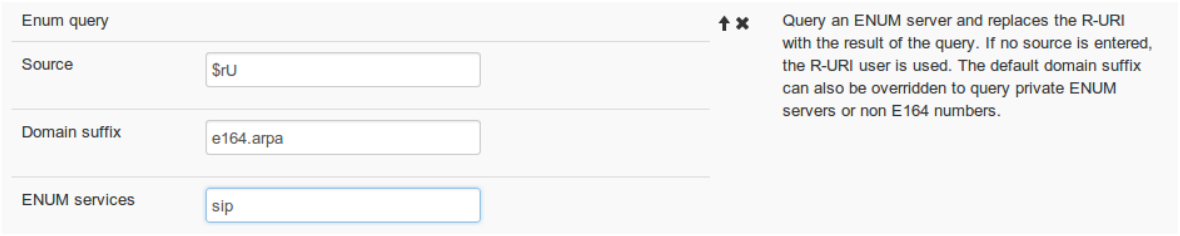


Fig. 79: Using Enum queries

By using a different domain suffix than the default one (e164.arpa), private enum servers can be queried. This is in fact the way ENUM is widely used – as of today, no public ENUM service with global coverage has emerged.

The result of the Enum query can be tested using the **Last Action Result** condition. If it returns true, the ENUM query returned a URI, false is returned otherwise. In case of success, the ENUM-returned URI has rewritten the request-URI and may be rewritten using the (**$rU**) replacement expression.

6.14 SIP-WebRTC Gateway

WebRTC is a relatively new protocol suite added to the VoIP technology that makes a telephone out of every capable web browser. As a result, users can click-to-dial a company representative, easily access video-telephony from within other web applications and receive calls from any web-browser, be it on their PC, smartphone or Internet café.

All of that while enjoying confidentiality widely available to consumers as never before in telephony's history. Both analog and digital telephony were inherently insecure, mobile telephony secured at least the wireless hop, yet rather weekly. SIP's security protocols, PGP, S/MIME and Identity (**RFC 4474**) desperately failed to be adopted. With WebRTC, we have proven web-based cryptographic protocols that just work!

The key missing piece for connecting Web clients to the SIP telephony is a SIP-WebRTC gateway – see the left-most element in the Figure *Integration of RTC, SIP and PSTN Networks using the RTC Gateway*. The gateway connects the populations of web users, SIP telephony users and traditional telephony users behind PSTN gateways. The gateway also provides a practical and yet fairly secure communication model: on the "internal" SIP-based side of the gateway, traditional IT practices for securing controlled networks can be used, while on the public Internet facing side proven cryptographic protocols are used. That is where the ABC SBC comes in: its border control instruments in combination with built-in RTC gateway allow to form a viable security model.

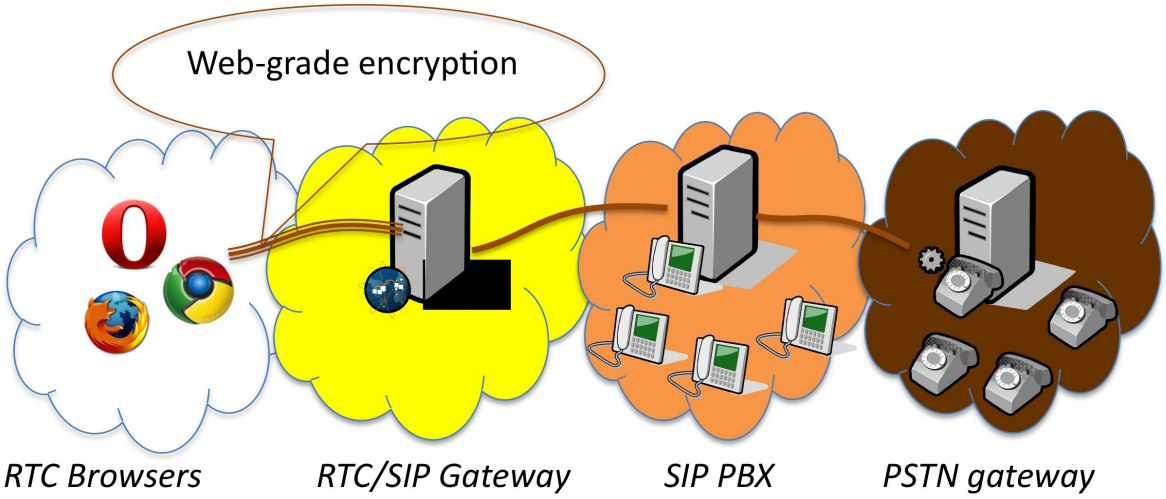


Fig. 80: Integration of RTC, SIP and PSTN Networks using the RTC Gateway

The gateway anchors signaling and media and performs translation between different standards for WebRTC and traditional VoIP, particularly security, codecs and signaling protocols as shown in Figure *WebRTC Gateway Protocol Stack*.

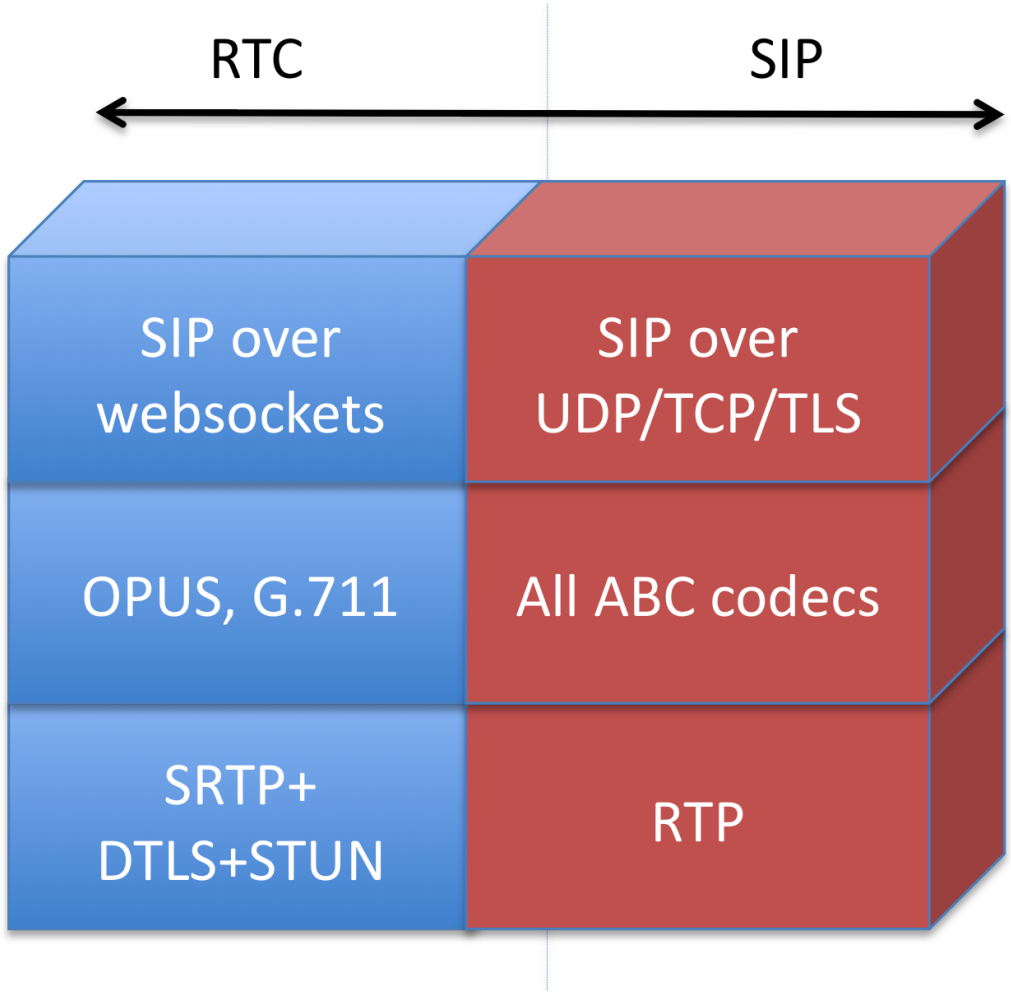


Fig. 81: WebRTC Gateway Protocol Stack

Integrating a gateway in a SIP network is fortunately straight-forward. When a SIP-WebRTC gateway is installed and configured to connect to an existing SIP services (PBX, public SIP service), WebRTC clients can immediately reach and be reached from the SIP service. The existing SIP service does not need to be modified at all – it treats WebRTC traffic from behind the gateway as regular SIP traffic.

The rest of this section is split in the following parts: brief introduction to the WebRTC protocols and network architecture is given in Section *WebRTC Network Architecture and Protocols*. Configuration of the gateway is explained in subsequent sections: *WebRTC Network Configuration*, *WebRTC Credentials Configuration*, and *WebRTC Rules Configuration*. Eventually we provide guidelines for starting an RTC gateway using the Amazon Elastic Cloud services in Section *Amazon Elastic Cloud Configuration Cookbook*. We offer several methods using either predefined configurations or using manual configuration, and starting a single gateway or a whole failsafe cluster. We also provide recommendations for starting a geographically-dispersed service.

If you plan to start the RTC gateway service in front of an existing SIP service rapidly, best proceed directly to the Section *Amazon Elastic Cloud Configuration Cookbook*.

6.14.1 WebRTC Network Architecture and Protocols

The WebRTC protocol suite for telephony specifies use of the following protocols:

- G.711 and OPUS (**RFC 6716**) for audio codecs. Opus is a lossy compression, low-delay, codec with constant and variable bitrate ranging from 6kbps to 510 kbps. G.711 is legacy PSTN audio codec at 64 kbps.
- VP8 (**RFC 6386**) for video codec. VP8 is an irrevocably royalty-free codec.
- SRTP (**RFC 3711**) for secure real-time media transmission.
- DTLS (**RFC 4347**) for keying. - SIP over Websockets (**RFC 7118**) as one of possible protocols for signaling. It is slightly aligned SIP using websockets as transport. It is particularly easy to translate to and from legacy SIP.
- ICE (**RFC 5242**), STUN (**RFC 5389**) and TURN (**RFC 6062**) for NAT traversal. STUN is a probing protocol that allows clients to detect how it is reachable over NATs. TURN is a STUN-based protocol that allows a client behind NAT to allocate a publicly reachable IP address from a server and tunnel traffic from and to it. ICE is methodology for finding the best combination of IP addresses to communicate between clients.

At the time of publication of this handbook, Firefox (version 23 and above) Chrome (version 28 and above), Opera (version 20 and above) and Safari (Preview, June 1017) were supporting this protocol stack and have demonstrated mutual interoperability. Several JavaScript applications[1] emerged that implemented signaling using SIP over websockets.

In the simplest scenario, two browsers can use the protocol stack to interconnect with each other. Most of this document is however concerned with the case when one party is using a WebRTC capable browser, and the other party is using a SIP phone or a PSTN phone behind a SIP gateway. This is the most complicated and also critical scenario because it connects the web telephony users to existing population of SIP users. The key component in this scenario is WebRTC-to-SIP gateway which translates signaling and media between the WebRTC and non-WebRTC SIP protocol stacks.

The WebRTC clients use the protocol stack is shown in Figure *RTCWeb Protocol Flows*. Initially the client registers itself to become reachable for incoming calls. It does so by sending a SIP REGISTER message over websockets. It is that simple.

[1] The JSSIP application is available under MIT License and can be obtained from http://jssip.net.

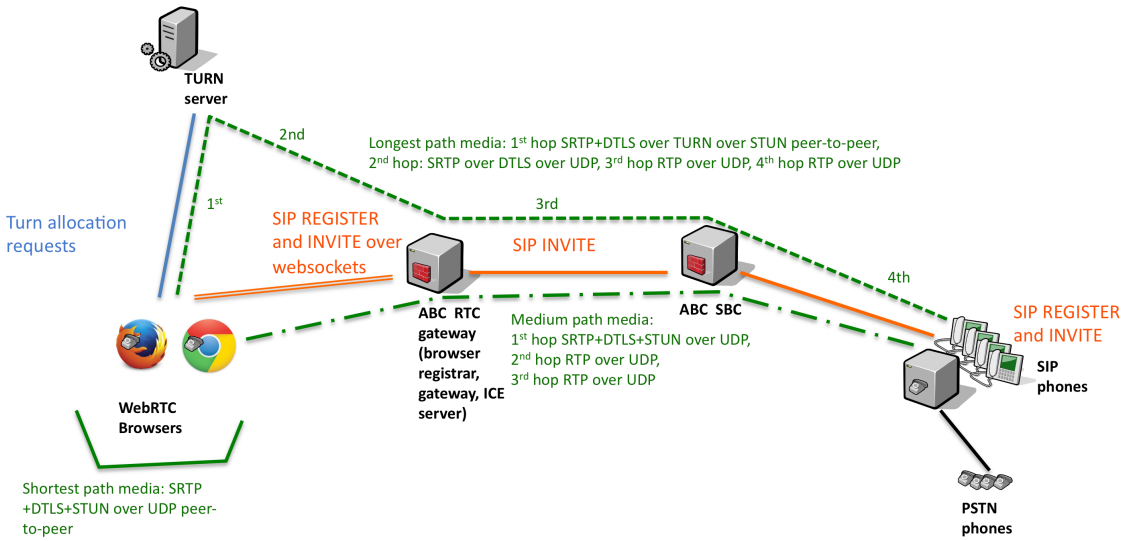


Fig. 82: RTCWeb Protocol Flows

When the browser user wants to make a call, it is a more complicated process. The browser starts the ICE process in which it learns IP addresses under which it can be reached. The IP addresses include the WebRTC client's own, its own as seen on the Internet and learned using the STUN protocol, or even a completely different IP address belonging to a TURN media-relay. When the browser initiates SIP signaling, it offers all IP addresses learned in the previous phase. After the called party answers the call, the client probes the IP addresses against the caller to chose the IP address with best IP connectivity. When the "best" IP address is chosen, an encryption session-key is generated using DTLS and media is exchanged using SRTP.

The actual media call-flow can vary depending on how the WebRTC application is configured and the actual call-by-call result of ICE connectivity checks. In a typical scenario deploying FRAFOS gateway, media is sent directly between the WebRTC client and the gateway. This is shown in the Figure *RTCWeb Protocol Flows* as the green dashed-dotted line. However, the WebRTC application can be also configured to communicate using a TURN server which introduces another hop to the media path. That's the dashed green line in the Figure. It can be for example useful if one wishes to relay media using TCP protocol. It can also occur that both call parties are WebRTC clients on the same subnet and media can flow the shortest-path between them – shown as solid line in the Figure.

However, in scenarios using the gateway the most practical client configuration choice is to limit ICE process to its own IP address. That eliminates gathering the STUN and TURN choices and greatly reduces "post-pickup delay", i.e. the period of time between when the caller answers and media can be actually heard and seen.

6.14.2 WebRTC Network Configuration

This subsection is about what components must be placed in the network and how they must be configured to enable working WebRTC call-flows. First, the following planning questions must be answered:

- do you want to enable NAT/firewall traversal using media over TCP? This may increase the NAT/firewall traversal success rate. If so an additional TURN server[2] must be introduced.
- which client do you want to use? The RTC-capable Web-browser alone includes the RTC engine but still needs an application that uses it. There are various commercial and open-source projects implementing the VoIP functionality, such as JSSIP. The ABC SBC comes with a JSSIP-based application for demonstration purposes. Note however, that FRAFOS does not support third-party client software. Keep in mind that the JavaScript code offered to WebRTC clients must include proper configuration of TURN and STUN servers.
- do you want to integrate the gateway functionality in an SBC or run it on a dedicated server? We suggest to use a dedicated server unless you have a good reason for tight integration. With a dedicated server, it is easy to discriminate WebRTC-to-WebRTC calls from WebRTC-to-RTC, apply different security logic to WebRTC clients, and avoid interference with legacy-SIP configuration.
- under which IP address and port number will be the websocket interface available? To enable websocket communication, you must configure an SBC interface and create a Call Agent linked to the interface. The interface configuration dialog is shown in Figure *Websocket Interface Configuration*. The most important element is "Interface type" which must be set to "websocket signaling". The Call Agent configuration is shown in Figure *Websocket Call Agent Configuration*. By setting its interface to the previously created websocket interface and its IP address to "All" (0.0.0.0/0), it captures every WebRTC clients communicating with the ABC SBC using websockets.

[2] A TURN server is not part of ABC SBC. A publicly available TURN server is available under from https://code.google.com/p/rfc5766-turn-server/

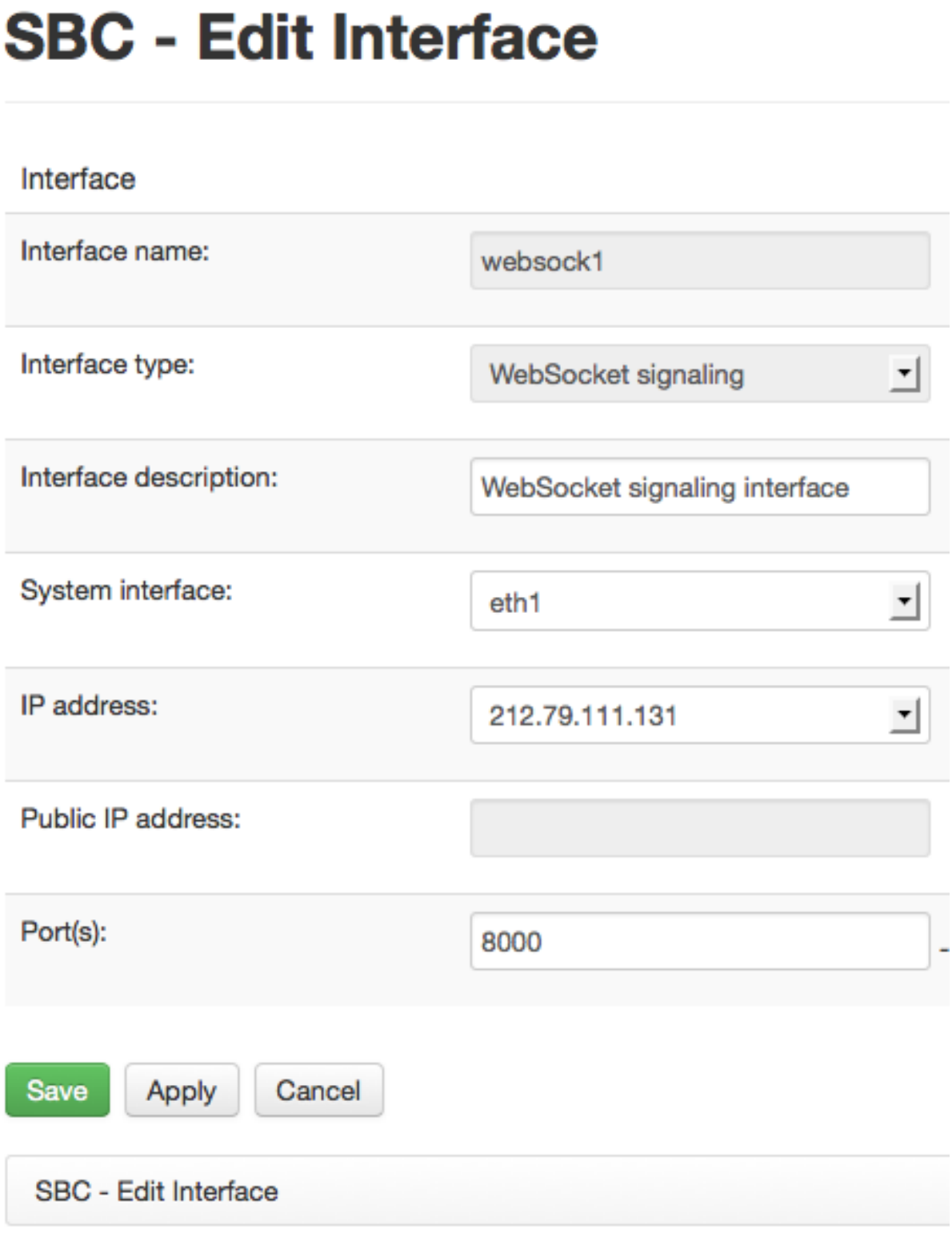


Fig. 83: Websocket Interface Configuration

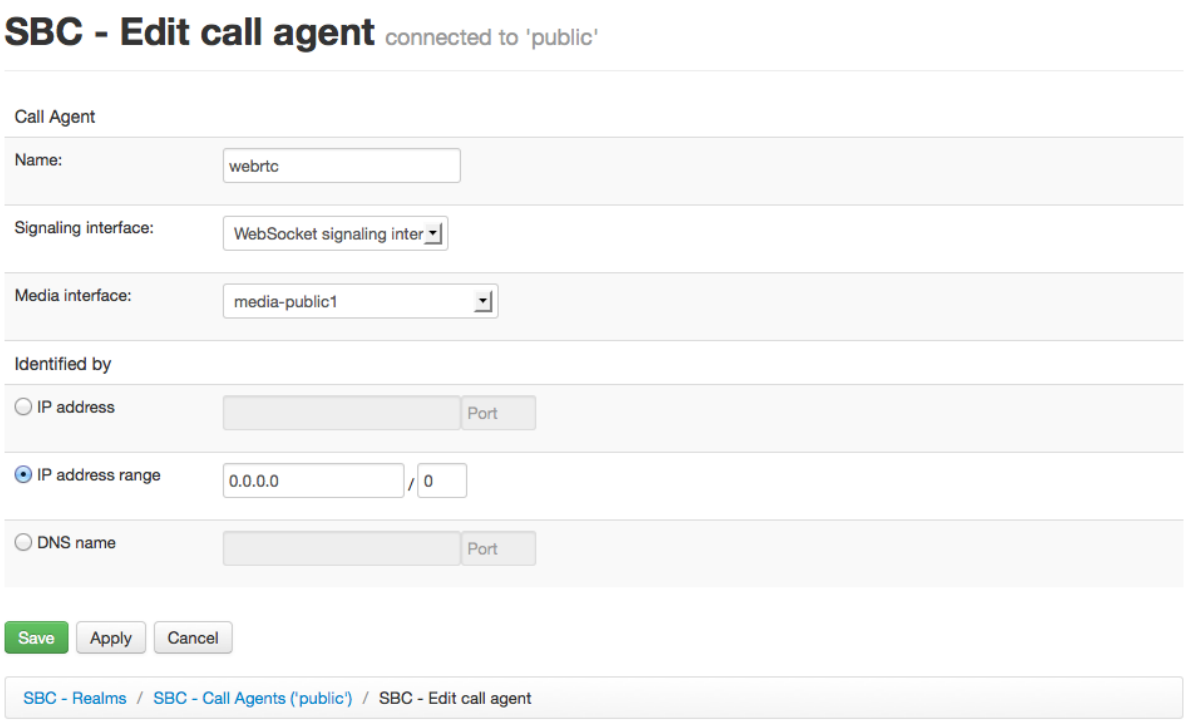


Fig. 84: Websocket Call Agent Configuration

6.14.3 WebRTC Credentials Configuration

Confidentiality of calls by encryption is one of the major WebRTC features. Fortunately, it is rather easy to configure. DTLS-SRTP is always enabled in current version of ABC SBC (in previous versions there was possibility to disable it). All other configuration options are optional. Such configuration is shown in Figure *SRTP Configuration Page*.

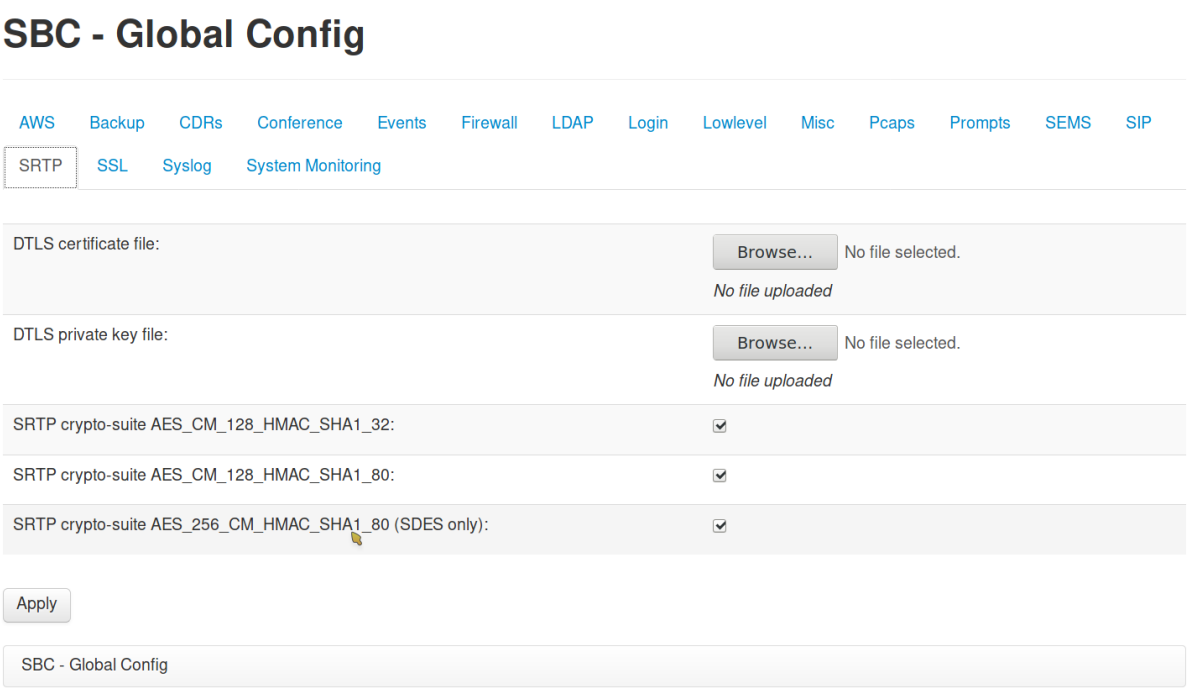


Fig. 85: SRTP Configuration Page

When no further options are selected, the ABC SBC creates ad-hoc self-signed credentials. A particular advantage of these is the length of resulting DTLS-SRTP packets will be bellow 1500-bytes packet length which is almost always certain to traverse networks without IP fragmentation.

If you prefer your own certificates, you must upload them using the "DTLS certificate file" and "DTLS private key file" global config options (located under Misc tab).

Note that some credentials may result in too long DTLS-SRTP packets. If they exceed the length of 1500 bytes, they will be most likely fragmented and may result in failure to set up media channel. This is almost certain if there are NATs along the communication path.

6.14.4 WebRTC Rules Configuration

The configuration of the rules for SIP-WebRTC gateway must address both generic SIP processing aspects, which is routing and NAT traversal, and then specific aspects of WebRTC interworking.

In this configuration example we assume topology shown in Figure *RTCWeb Protocol Flows*, two types of calls: WebRTC-to-RTC and RTC-to-WebRTC, and media flowing through the ABC SBC along the dash-dotted green line.

The SIP routing flow is rather simple in this scenario: every call coming from the WebRTC Call Agent (i.e. over the websocket interface) will be routed to a SIP PBX, and reversely every call coming from the PBX will be routed to RTC browsers using websockets. The routing configuration is shown in Figure *SIP-WebRTC Gateway Routing Rules*.

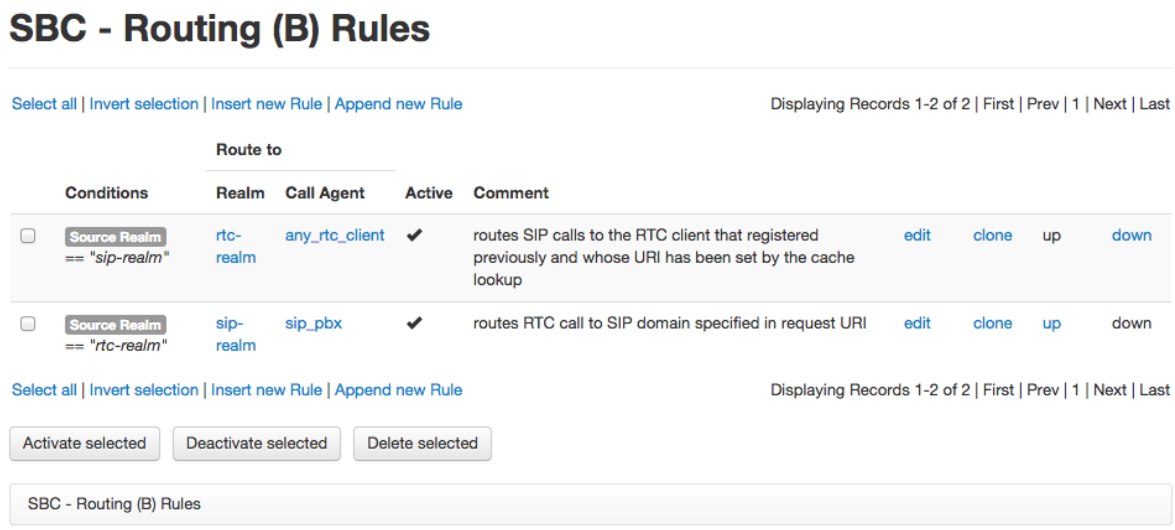


Fig. 86: SIP-WebRTC Gateway Routing Rules

The task of A and C rules is to anchor media to itself and to determine when to convert calls from RTC to SIP and vice versa. Therefore we create two realms: one for RTC clients and one for SIP clients. For each of them, we create one Call Agent that captures all traffic from/to any IP address flowing through the websocket and SIP interface respectively. The actions are configured to accommodate the following policies :

Realm	Direction	Policy (Actions)
RTC	A-rules	• enforce frequent re-REGISTERs to keep persistent TCP connections for websockets alive (REGISTER throttling) • cache registrations to forward SIP calls for RTC clients properly (Enable REGISTER caching) • fix NAT bindings (Enable Dialog handling) • anchor media, offer ICE and RTC Feedback to RTC clients (Enable RTP Anchoring)
RTC	C-rules	• anchor media (Enable RTP Anchoring) • enforce SRTP using DTLS keying (Force RTP/SRTP)
SIP	A-rules	• lookup registered RTC user, decline the call if offline (Reply to request with reason and code, Retarget R-URI from cache (alias))) • anchor media, don't offer ICE to SIP callers (Enable RTP Anchoring)
SIP	C-rules	• anchor media (Enable RTP Anchoring) • enforce plain RTP on the way to the SIP Call Agent Force RTP/SRTP

We have met most of the rules in previous sections: driving re-registrations high to keep transport-layer connections alive, caching registrations, fix bindings and anchor media. Now we need to include the specifics of SIP and RTC interworking. SIP calls towards RTC clients must appear RTC-capable, i.e. they must offer SRTP encryption, ICE connectivity checks and RTC feedfack. Reversely, the RTC calls to SIP must be transformed to plain RTC.

The "Force RTP/SRTP" action determines if plain RTP or SRTP is used for a call. When this action is placed in C-rules, it converts media for the called party into the enforced protcol. When SRTP is chosen, one must set an additional option: the keying protocol. Only DTLS makes sense for RTC. In our example we convert all media traffic towards SIP devices by placing "Force RTP" in SIP realm's C-rules. Analogically we convert all media traffic towards RTC clients by placing "Force SRTP" in RTC realm's C-rules. The "Force SRTP" action is using "DTLS" as the keying option because that's the keying protocol standardized for use with RTC.

One could also use the "Force RTP/SRTP" action in A-rules: here however it only determines if the caller's SDP offer complies to the enforced preference and rejects the call otherwise. We are not using this kind of admission policy in our example.

The other options specific to the RTC interworking use-case are specific to how we anchor media. We need to make sure that RTC clients relying on ICE will receive proper STUN answers for their connectivity checks

towards the built-in media relay and also RTC feedback. Therefore, we turn the options "offer ICE" and "offer RTCP feedback" on in the media anchoring action for both RTC A-rules and C-rules. The A-rules make sure that incoming RTC call offers obtain ICE and RTC/F capable answers, the C-rules ensure that SDP offers towards the RTC clients will be also ICE and RTC/F capable.

The resulting configuration is shown in Figures *Configuration of RTCWeb Rules for RTC Realm* and *Configuration of RTCWeb Rules for SIP Realm* for the RTC and SIP realm respectively.

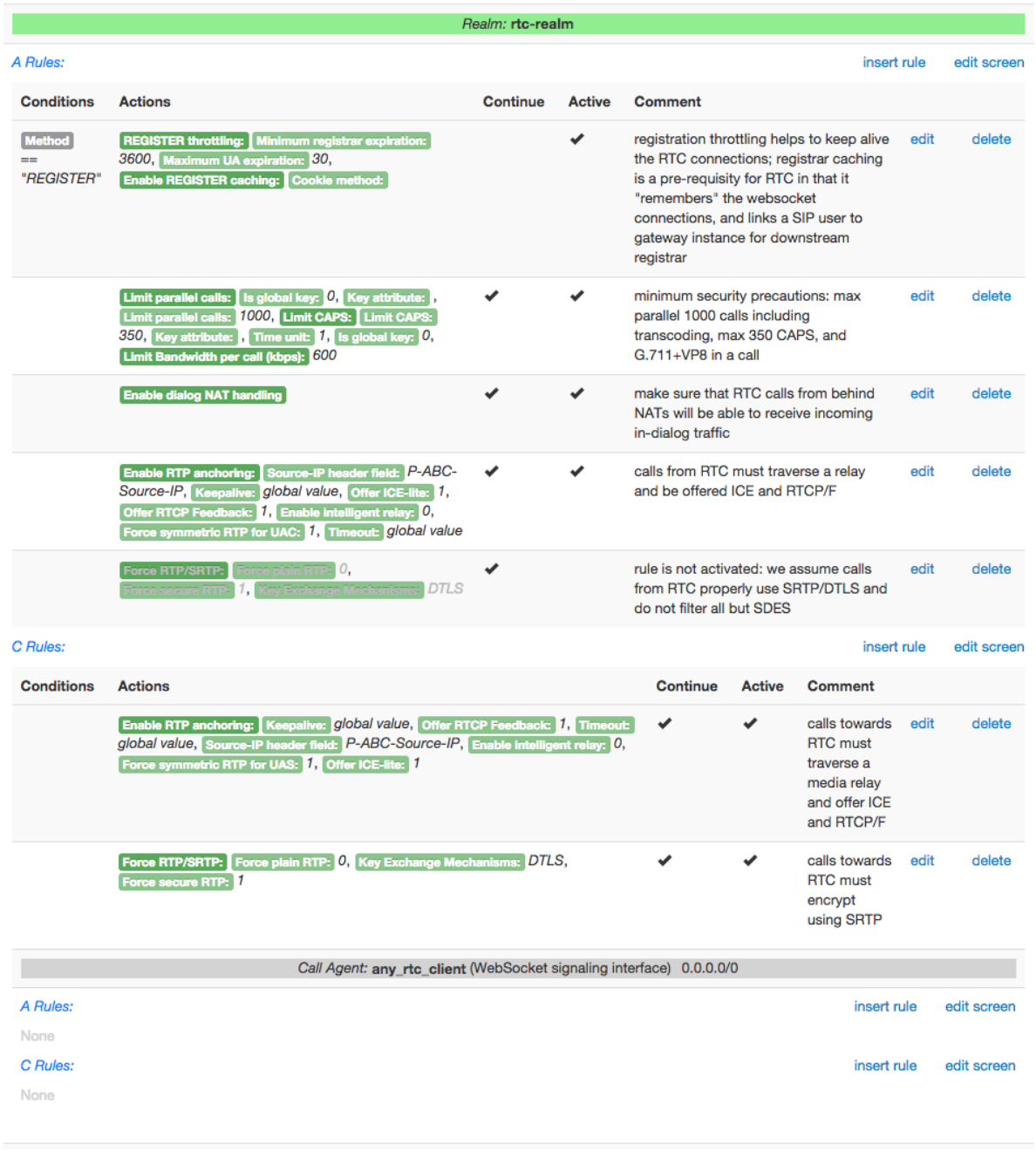


Fig. 87: Configuration of RTCWeb Rules for RTC Realm

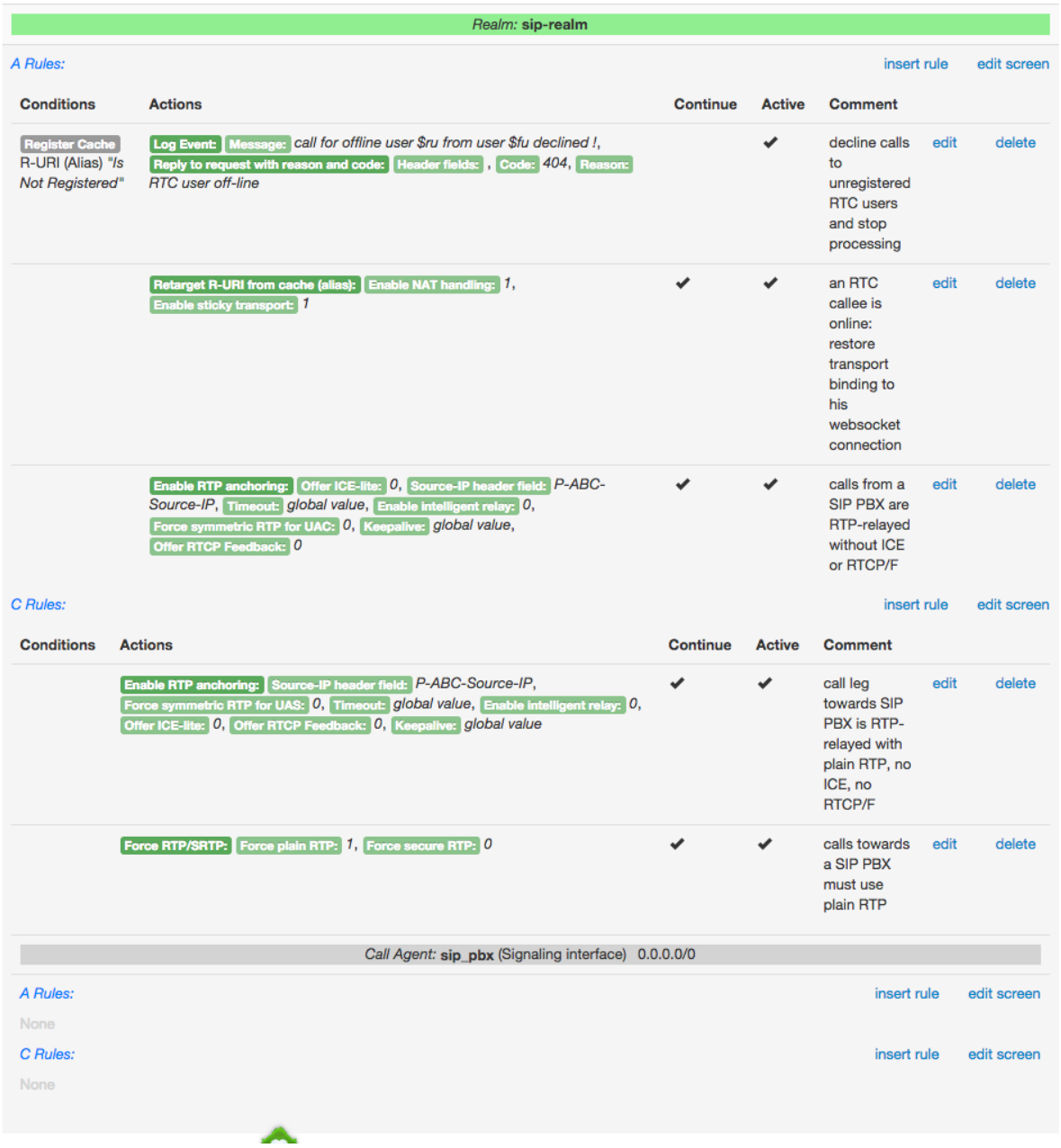


Fig. 88: Configuration of RTCWeb Rules for SIP Realm

Note that this configuration works even if two WebRTC clients connect to each other through the gateway. However the WebRTC-to-RTC conversion and forwarding to the SBC still takes place resulting in an WebRTC-to-RTC-to-WebRTC loop,as shown in Figure *The WebRTC-to-WebRTC Lopback*.

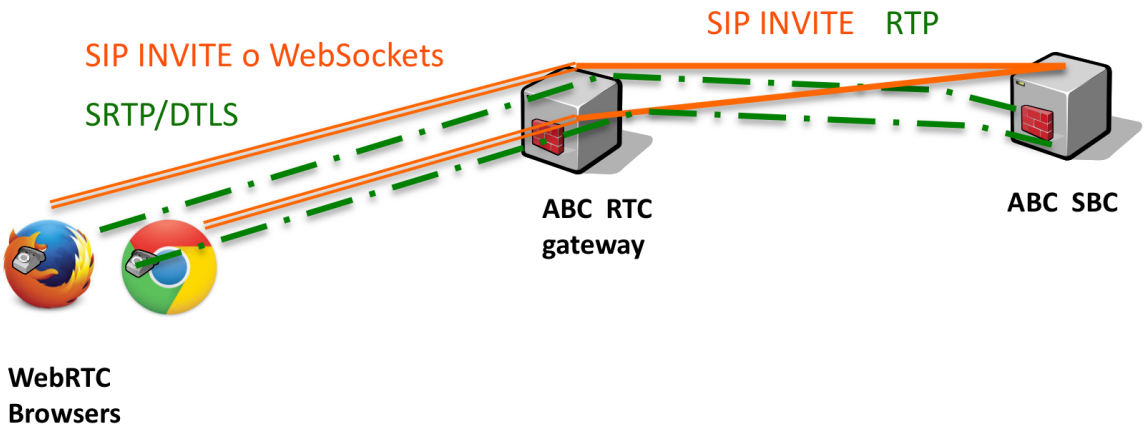


Fig. 89: The WebRTC-to-WebRTC Lopback

Optionally it may be useful to manage codec negotiation. For example, one could blacklist G.711 in favor of OPUS, if there are SIP clients that can speak the codec. Or video could be stripped off, if there is no support for royalty-free VP8 codec. Note though that if codecs are stripped too agressively, a SIP user agent may fail to interoperate and return a 488 in UAS or an immediate BYE in UAC role.

6.14.5 WebRTC Interoperability Recommendations

The WebRTC standard and implementations are relatively new and as result degree of interworking largely depends on network configuration and used client. Unfortunately interoperability is still changing with every new version of WebRTC stack and the clients built upon it.

Network complications typically arise when there is a "middlebox", an Application Layer Gateway (ALG) or an HTTP proxy in the path. This sort of network equipment manipulates HTTP traffic in a way that may impair interoperability. If the middlebox cannot handle the websocket extention of the HTTP protocol, signaling connection will fail. Therefore the default transport protocol for SIPoWebsockets is TLS.

WebRTC application complications typically arise when the application has imperfect support for the SIP protocol running on top of websockets, and/or changes its behaviour with a new software version. ***We urge our customers to test extensively the client application before intial deployment of a WebRTC service AND during an update to a newer version.***

The most "fluid" inteoperability difficulty is continuous changes to the WebRTC protocol stacks hidden insider the browsers. Almost with every browser release, some minor changes appear that impair interoperability. Until the environment becomes more stable, typical reaction is reverse analysis of the new interop behaviour and using ABC SBC mediation features to address it. For example, Chrome browsers Version 39.0 and higher are known not to handle "early media" correctly. The ABC SBC configuration allows to mediate "183 early media" into regular "180 ringing" as shown in Figure *WebRTC Mediation Example*.

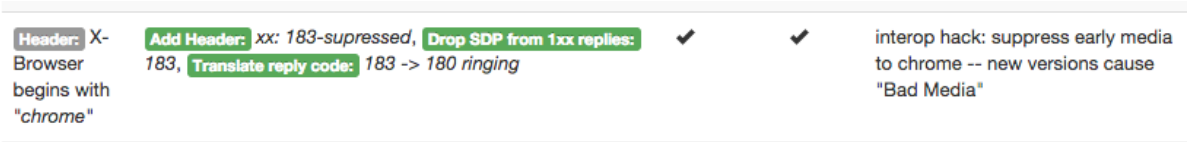


Fig. 90: WebRTC Mediation Example

In summary, while the industry is converging to a solid level of interoperability, thorough effort during initial and regression tests is highly recommended.

6.15 Amazon Elastic Cloud Configuration Cookbook

An easy way to start and run an RTC gateway is "off a cloud", i.e. on a hosted platform, without purchasing and operating own physical infrastructure: computers, racks, disks and IP connectivity. A single click is enough to start a service, of course as long as you keep paying for the cloud services. Whereas there are many "cloud platforms", we focus on running the RTC gateway on the Amazon Web Services (AWS) platform in this section.

The AWS platform is a mature system that allows, among many other useful things, to start and run pre-built virtual machines, load-balance traffic among them, monitor their health and scale the infrastructure up and down to be on par with user load. Applications, RTC-to-SIP gateway in our case, come pre-installed ready-to-start in form of a virtual machine image, called AMI (Amazon Machine Image).

The FRAFOS RTC gateway installation is preconfigured to address a simple yet useful scenario: add RTC connectivity to a running SIP PBX service. Once started, the RTC gateway passes RTC registrations and calls coming from RTC clients down to the PBX(s). Reversely, the gateway routes calls coming from the SIP PBX(s) to the previously registered RTC browsers. No further configuration is needed.

The following subsections describe how to start the RTC-to-SIP gateway service off the amazon platform. We offer you several ways to do the same: the easiest is launching a cluster using Cloud Formation template. This way you create a load-balanced scalable infrastructure by pressing a button without any further knowledge of how the components must be configured. If you want to understand in more detail how the gateway works, you can launch a single-instance service and/or configure it in detail step by step.

We suggest you explore our demo site https://go.frafos.com. It includes additional information about use of AWS and WebRTC technologies, including live services and ready-made demo AMIs and Cloud Formation Templates. These can be launched by a single click without any need for further configuration. Note that the demo versions have a 90-seconds limitation to maximum call duration.

6.15.1 Before you Start: Prerequisites and Important Warnings

Before you start, you shall have the following:

- Amazon Web Services (AWS) account. Note that the accounts come with several service plans charged at different levels, and credit card number and a telephone must be ready to verify identity and payment. Go to http://aws.amazon.com to sign up.
- AWS Elastic Cluster SSH keypair. This is important to be able to administer the virtual machines remotely. If you haven't created or uploaded one, do so under "EC2→Keypairs". If you want to start the services in multiple regions, make sure that you have a keypair for every region before you start.
- Amazon Machine Image (AMI) with the RTC-2-SIP gateway from FRAFOS. You will find the right one for your geographic region on our experimental webpage, https://go.frafos.com/.
- RTC-enabled browser for testing. Latest version of Chrome has been tested by FRAFOS to play well, yet there are other implementations as well.
- Optional: Publicly available SIP service and a SIP account. You need to have a SIP URI and password with a SIP service to be able to make calls through the RTC-to-SIP gateway. Otherwise you can only make anonymous calls.
- Optional: a DNS name under which your RTC-to-SIP gateway will be reachable

To begin visit our experimental webpage https://go.frafos.com/. The webpage contains predefined links to available AMIs that allow you to launch quickly.

Note: IMPORTANT: USE OF AMAZON WEB SERVICES WILL INCURE ADDITIONAL COST. ALL DATA CREATED AND STORED ON AN INSTANCE SUCH AS PROVISIONED TABLES, ABC RULES, CONFIG-

URATION PARAMETERS, LOG FILES AND MORE REMAINS ON THE INSTANCE AND WILL BE LOST UPON INSTANCE TERMINATION.

6.15.2 Quick Start Using Cloud Formation

The ultimately fastest way to launch your service is using amazon's Cloud Formation. The Cloud Formation amazon.com service is used to quickly start a whole network based on a description included in a template. The template includes information about virtual instances, how to scale them up and down, how to spread the load across them using a load-balancer, and what firewall policy to use to filter IP traffic: quite some work if administrator was setting all of this up manually.

FRAFOS has created a starter template to be used to start a fail-safe cluster of one-to-four gateways behind a load-balancer. The template is available on our site, https://go.frafos.com.

During the process you will be prompted for very few parameters. Their scope can change as we keep developing the template and for most cases they are best served by leaving them to their default values. The only required parameter you must set is the name of your SSH key. Once you start the cloud formation process, it takes several minutes until it completes. After the stack is launched, you will have one load-balancer and one to four gateways running behind it. A URI shown upon completion of the cloud formation process will allow users to download a demo JavaScript application and start using the service. Sometimes you may need to be patient for a couple of minutes until the service is really "warmed up".

When trying to place your first phone call, you may for example try to call sip:music@frafos.net. When opening the web-page, allow the browser to accept self-signed certificate and use your microphone and camera.

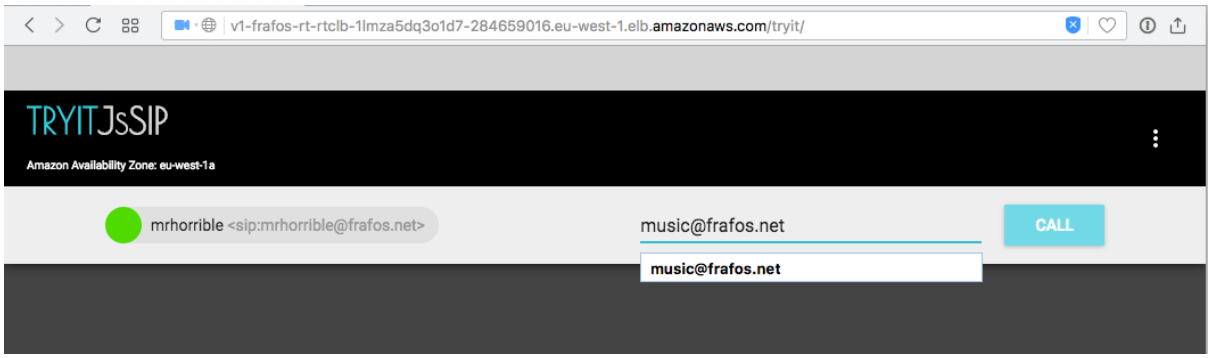


Fig. 91: Screenshot: First Browser Call to music@frafos.net

Also you can try out the built-in audio conferencing bridge by dialing an 8-digit number prefixed with *. Anyone calling the same address will appear in the same conferencing room.

As the next steps, you can follow the links that show in the Cloud Formation Output window: a WebRTC web telephony application and the ABC Monitor (use sbcadmin username and default password). You can also administer the actual instances by going to their web address "https://IP/", username "sbcadmin" and password equal to instance ID. For example, you can review rules that remove video streams between WebRTC and legacy SIP to allow at least audio where video signaling often fails, or look at the dialing plan for the on-board conferencing.

6.15.3 Quick Start: Launch Single Instance

If beginning with a cluster may appear too heavy start, one can also start a single RTC gateway instance instead. This can be done also from our site, https://go.frafos.com.

During the process you will be prompted for instance type, detail, used storage and security group. Choose an instance type with at least 2GB RAM and leave everything else except the Security Group to default. The security group must be set to permit the following flows from 0.0.0.0/0:

- TCP/5060-5069 — SIP service
- UDP/5060-5069 — SIP service
- TCP/443 - web user interface (/) and WebRTC Demo App (/tryit/)
- TCP/22 — secure shell
- UDP/10000-11000 RTP media

Eventually chose an existing or create a new key-pair and store the private key securely.

Once the virtual machine is up and running, you can access its trial application by going to its address using https://PUBLIC_IP/tryit/ or to its administrative interface using the https://PUBLIC_IP/. The administrative username is "sbcadmin", the password is the ID of your amazon instance. You can also access remote shell if you login using the private part of the AWS SSH key:

```
$ ssh -i .ssh/frafos-aws-keypair.pem -l ec2-user 54.171.123.109
```

If you would like to use additional AWS features that the instance supports, CloudWatch and System Manager, you must enable an instance role that permits these. An easiest way to do so is to create an AWS/EC2 role with predefined permissions "EC2 Role for Simple Systems Manager" (AmazonEC2RoleforSSM) and attach it to the instance.

6.15.4 Updating License

On Amazon Cloud there is an easy way to install centrally a license file that is then used by all newly started ABC SBC instances. This is practical when you upgrade to a feature-richer license and do not want to configure the license individually in every new instance. The license is then used by both instances that are newly started individually as well as via Cloud Formation and AutoScaling. You only need to make sure the license file matches the AMIs you are using.

After obtaining the license file from FRAFOS support, all you need to do is to enable instance's access to Systems Manager (see http://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-access.html#sysman-configuring-access-role) and put the license in a parameter with a well-known name in the Parameter Store. The Parameter Store is located under the EC Dashboard under "System Manager Shared Resources → Parameter Store". The parameter name must be "/abcsbc/license" as shown in the screenshot bellow.

Note that setting this parameter does not affect running instances, only applies to the AWS Region for which you provisioned it, and must include a license specific to the AMIs you are using.

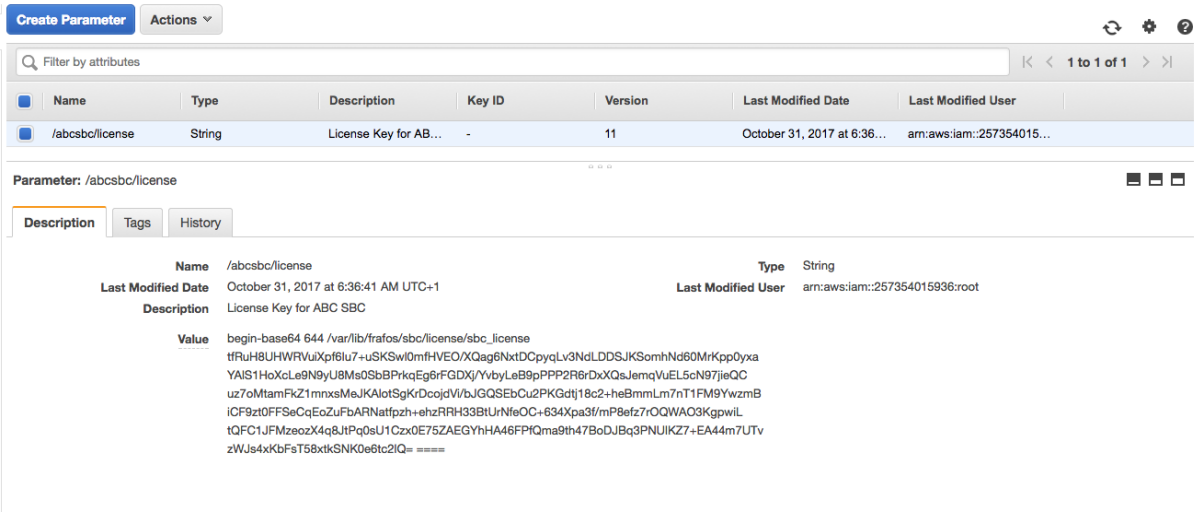


Fig. 92: Screenshot: Setting License in Amazon Parameter Store

6.15.5 Introducing Geographic Dispersion

Introducing geographic redundancy and dispersion may be useful to become resilient against regional disasters and/or decrease VoIP latency. Latency may have major impact on quality of service. For example if an American user accesses a European RTC gateway to reach an American SIP PBX, media will travel across the Atlantic back and forth, resulting in noticeable latency and QoS degradation.

Fortunately there is an easy-to-manage way with AWS to build up geographic redundancy for both individual instances and whole clusters. All that needs to be done is creation of the instances or whole stacks as described in previous subsections multiple times in different regions, and linking their addresses to a single latency-routed DNS name. That is a particular feature of Amazon Route53 DNS service, that returns the lowest-latency IP address associated with a DNS name.

We experimented with this amazon feature and confirmed significant latency savings. In our example, we created two instances, one located in Ireland, the other in California. We create CNAME records “”eu.areteasea.com” and “us.areteasea.com” for them. Eventually we created the latency-routed global DNS name entries “world” for both regions, as shown in Figure *Screenshot: Creating DNS Latency-based Routing Records*.

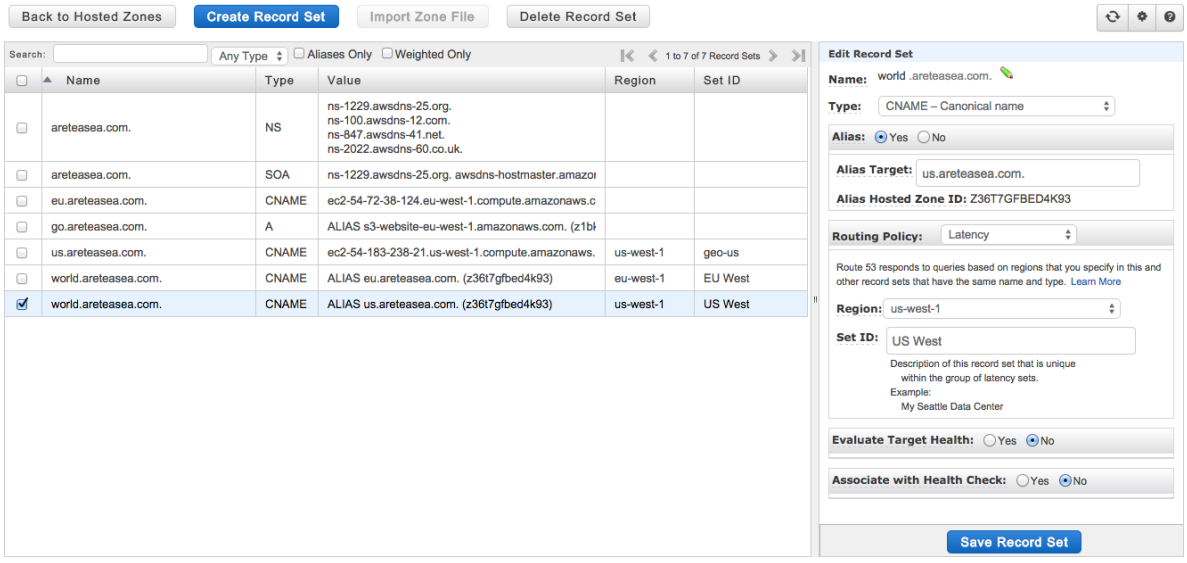
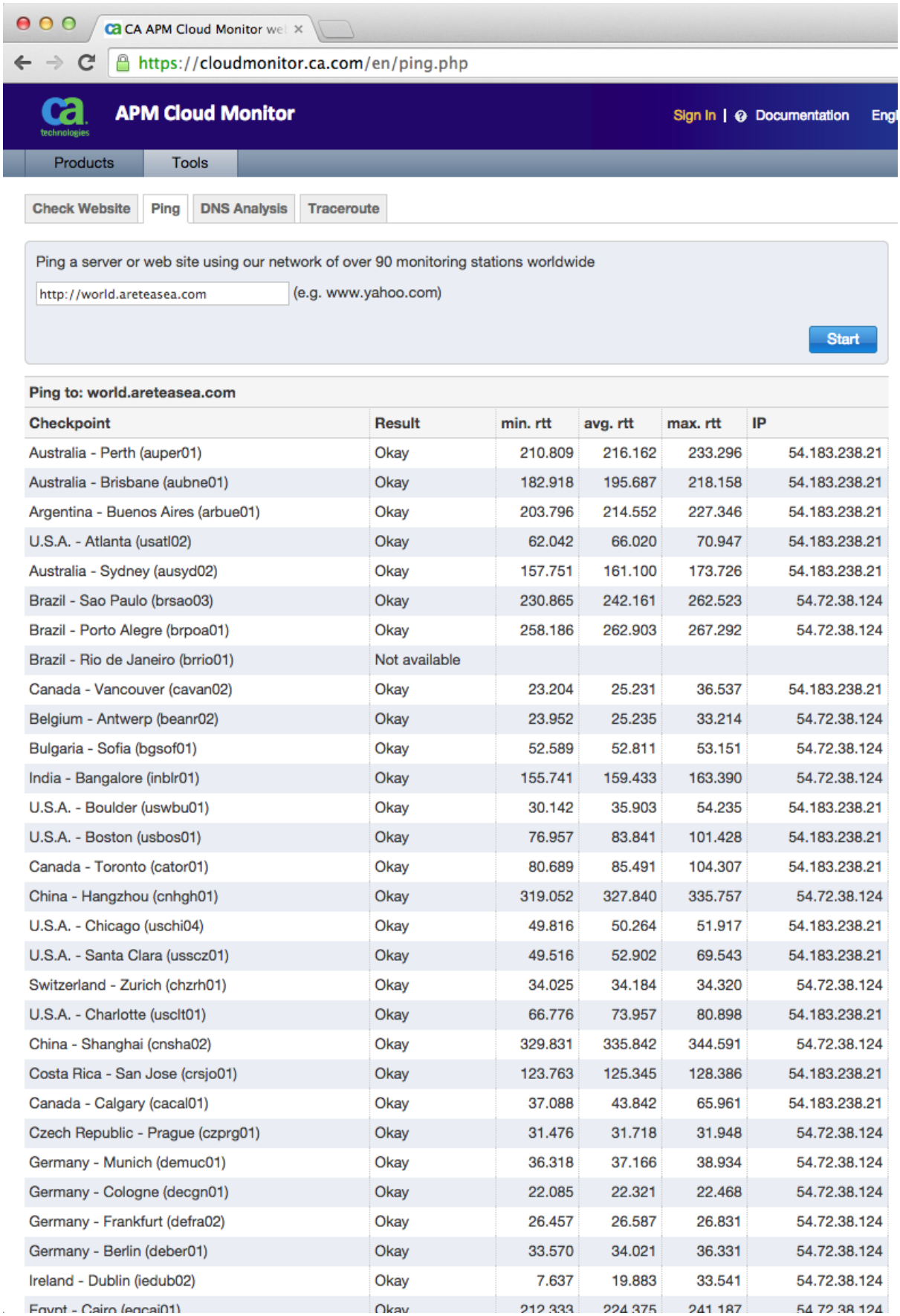


Fig. 93: Screenshot: Creating DNS Latency-based Routing Records

Clients trying to open up a connection to "world.areteasea.com" resolve this DNS name to different IP addresses depending on where they ask from.

One can easily verify the outcome by using services like Cloud Monitor. (http://cloudmonitor.ca.com) Results shown in Figure *Latency Measurements for Multiple Sites Served by Route-53 Latency-Routing* prove that proximity makes a difference. Clients in geographic proximity of the two sites feature minimum latency bellow 50ms: US from California to Illinois show 30 to 50ms, Western Europe shows 24-37 ms, Ireland 8 ms. Clients located out of served continents have significantly higher latency, starting with 180ms for Australia, slightly above 200ms for Argentina and Egypt, and peaking with 329 ms in China – values that make VoIP quality poor.



Ping to: world.areteasea.com

Checkpoint	Result	min. rtt	avg. rtt	max. rtt	IP
Australia - Perth (auper01)	Okay	210.809	216.162	233.296	54.183.238.21
Australia - Brisbane (aubne01)	Okay	182.918	195.687	218.158	54.183.238.21
Argentina - Buenos Aires (arbue01)	Okay	203.796	214.552	227.346	54.183.238.21
U.S.A. - Atlanta (usatl02)	Okay	62.042	66.020	70.947	54.183.238.21
Australia - Sydney (ausyd02)	Okay	157.751	161.100	173.726	54.183.238.21
Brazil - Sao Paulo (brsao03)	Okay	230.865	242.161	262.523	54.72.38.124
Brazil - Porto Alegre (brpoa01)	Okay	258.186	262.903	267.292	54.72.38.124
Brazil - Rio de Janeiro (brrio01)	Not available				
Canada - Vancouver (cavan02)	Okay	23.204	25.231	36.537	54.183.238.21
Belgium - Antwerp (beanr02)	Okay	23.952	25.235	33.214	54.72.38.124
Bulgaria - Sofia (bgsof01)	Okay	52.589	52.811	53.151	54.72.38.124
India - Bangalore (inblr01)	Okay	155.741	159.433	163.390	54.72.38.124
U.S.A. - Boulder (uswbu01)	Okay	30.142	35.903	54.235	54.183.238.21
U.S.A. - Boston (usbos01)	Okay	76.957	83.841	101.428	54.183.238.21
Canada - Toronto (cator01)	Okay	80.689	85.491	104.307	54.183.238.21
China - Hangzhou (cnhgh01)	Okay	319.052	327.840	335.757	54.72.38.124
U.S.A. - Chicago (uschi04)	Okay	49.816	50.264	51.917	54.183.238.21
U.S.A. - Santa Clara (usscz01)	Okay	49.516	52.902	69.543	54.183.238.21
Switzerland - Zurich (chzrh01)	Okay	34.025	34.184	34.320	54.72.38.124
U.S.A. - Charlotte (usclt01)	Okay	66.776	73.957	80.898	54.183.238.21
China - Shanghai (cnsha02)	Okay	329.831	335.842	344.591	54.72.38.124
Costa Rica - San Jose (crsjo01)	Okay	123.763	125.345	128.386	54.183.238.21
Canada - Calgary (cacal01)	Okay	37.088	43.842	65.961	54.183.238.21
Czech Republic - Prague (czprg01)	Okay	31.476	31.718	31.948	54.72.38.124
Germany - Munich (demuc01)	Okay	36.318	37.166	38.934	54.72.38.124
Germany - Cologne (decgn01)	Okay	22.085	22.321	22.468	54.72.38.124
Germany - Frankfurt (defra02)	Okay	26.457	26.587	26.831	54.72.38.124
Germany - Berlin (deber01)	Okay	33.570	34.021	36.331	54.72.38.124
Ireland - Dublin (iedub02)	Okay	7.637	19.883	33.541	54.72.38.124
Egypt - Cairo (egcai01)	Okay	212.333	224.375	241.187	54.72.38.124

Fig. 94: Latency Measurements for Multiple Sites Served by Route-53 Latency-Routing

After we had disabled the European site, all sites began to be served by the Californian server and we observed increase in minimum latency of European clients up to 160-180 ms, i.e. by about 130 ms! Therefore we recommend anyone serving global user population to consider establishing presence in multiple amazon's availability zones.

6.15.6 Monitoring the Autoscaling Cluster Using CloudWatch

Once the cluster is up and running, it may be worthwhile to experiment with its autoscaling behaviour and monitor how the cluster reacts to varying load. There are various ways how you can observe the status of the cluster using Amazon's CloudWatch facility. The CloudWatch facility collects data from all related instances and load-balancers, aggregates it for the whole autoscaling groups and triggers alarms if some critical values are exceeded. The collected data, how it is aggregated and when it triggers autoscaling alarms is part of the CloudFormation template definition, so if you started the cluster using the template it is already in place. By default, the autoscaling alarms add a new instance when it the average CPU load in the cluster exceeds 80% for several minutes, and remove an instance if it drops bellow 60%.

The interesting data you can observe include the event-by-event history under "EC2 → Autoscaling Group → Scaling History", details of autoscaling alarms in the CloudWatch Console, and graphs showing the cluster changes along a timeline are also found in the CloudWatch console. The rest of this section shows typical autoscaling situations and how you can inspect them using these monitoring facilities.

The first Figure *Screenshot: Scaling History* shows example of scaling history. We interpret it in time order from bottom up. Initially when the Autoscaling process started it launched the first instance at 12:34. Because we kept the machine busy, some seven minutes later at 12:40 the Autoscaling process chose to reinforce the cluster. It increased the desired capacity to 2 and launched a new instance. Then we started reboot of an instance to simulate a failure. The ELB checks detected the unresponsive instance at 12:48, terminated it, and started a new one. Eventually we relaxed the load, the low-CPU alarm was triggered in response to which the Autoscaling process reduced cluster size back to one.

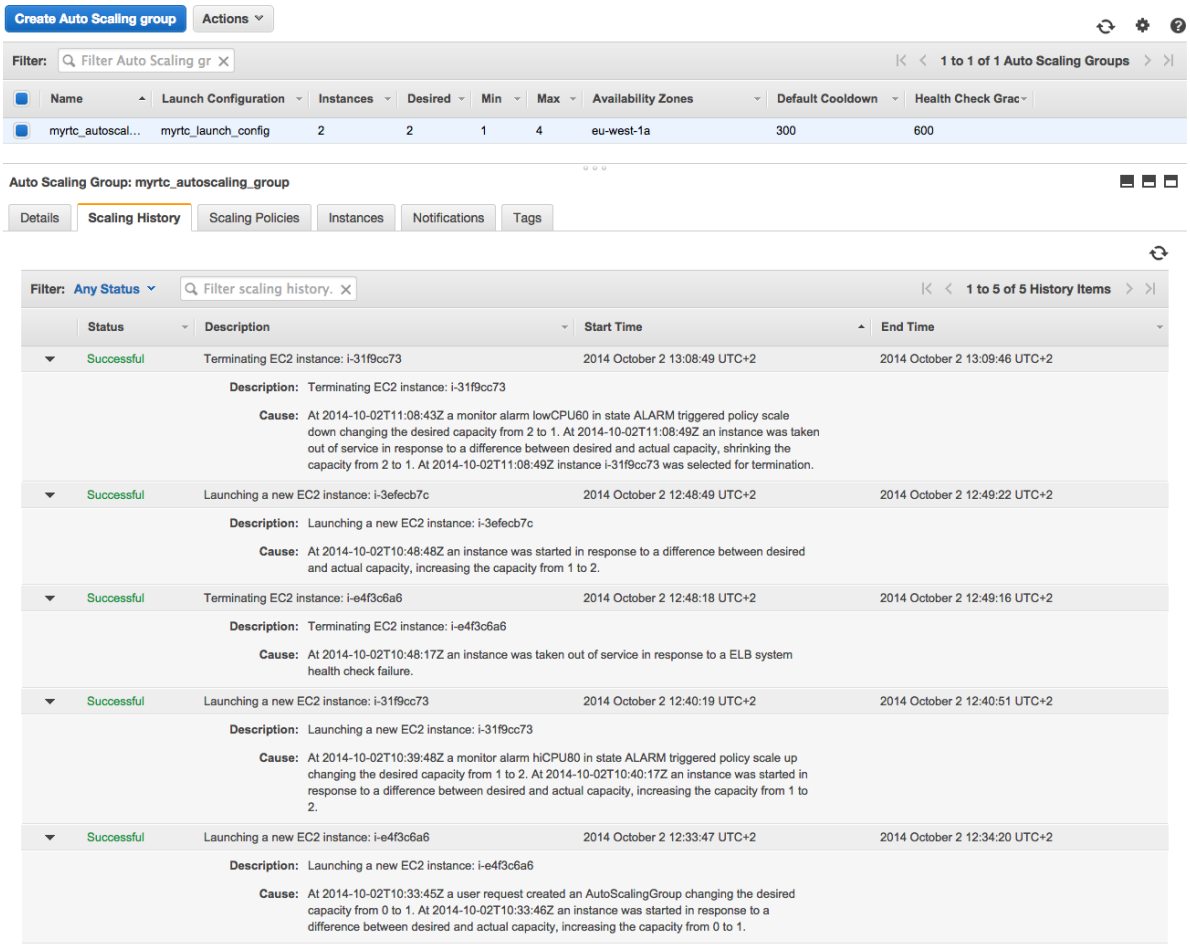


Fig. 95: Screenshot: Scaling History

The next Figure *CloudWatch Screenshot: Low Load Alarm* shows details of a CloudWatch autoscaling alarm. It displays a situation when cluster began to be idle after a period of congestion and an alarm is raised to scale the cluster down. The autoscaling process will remove an instance in response to this alarm.

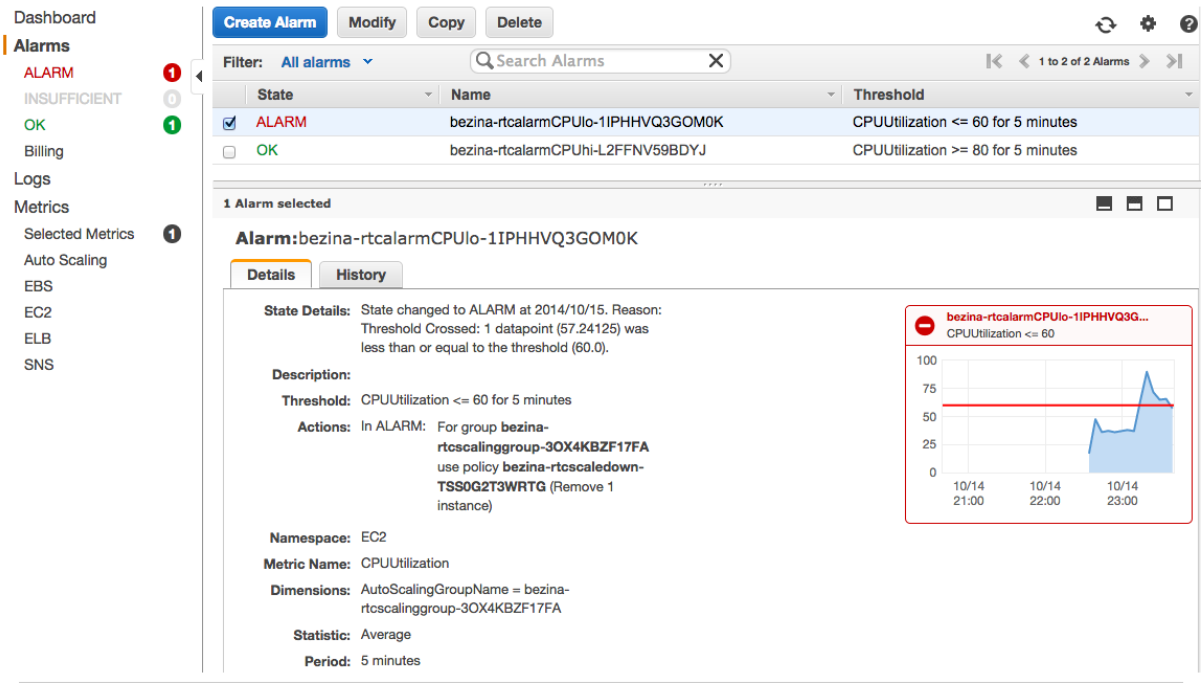


Fig. 96: CloudWatch Screenshot: Low Load Alarm

Development can be show in different time-scales using CloudWatch graphs. Figure *CloudWatch Graphs: Correlation of Cluster CPU Load and Autoscaling* shows how detection of overload and idle conditions affect cluster size along the time axis. There are three lines in the graph: the orange line shows average CPU load in the cluster. The autoscaling assessment of needed capacity is shown using the blue-line and the actual number of available instances is shown using green line. The CPU-load-line leads the changes: it must remain for a period of time above the threshold of 80% until the auto-scaling process determines to increase the target capacity. It then takes some time again until the capacity is ready: a new instance must be launched, detetected as ready and included in the load-balancer's distribution list. Therefore the green line legs behind the blue-line, and the blue-line always legs behind the orange-line.

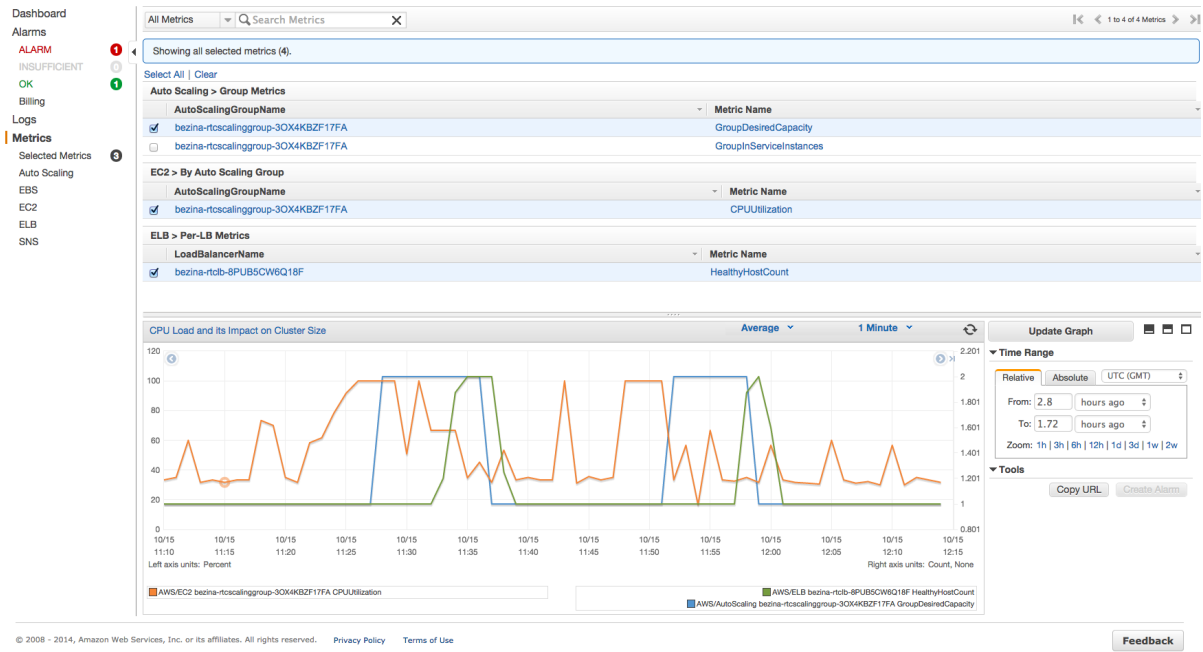


Fig. 97: CloudWatch Graphs: Correlation of Cluster CPU Load and Autoscaling

6.15.7 Performance Recommendations

In virtualised cloud environments, performance can vary significantly due to the "sharing" nature of these environments. It is therefore advisable to choose properly dimensioned computing instances. Amazon offers several types of "instance types" that very in various performance aspects. The instance type vary by region and change over time. Current offering is available on the amazon webpage http://aws.amazon.com/ec2/instance-types/.

For minimum density trials, the 2GB RAM T2.small instance type is sufficient. This instance allows very little CPU capacity in short bursts. However if the allowed burst is exceeded, the virtual machine will slow down to an extent that it stalls. Experiments with onboard conferencing have shown that a single conference with more than three participants already brings the machine to stalling.

For predictable performance, you will need a Fixed Performance Instance (FPI) type.

In the mainstream case, when media anchoring is enabled and there is neither transcoding nor encryption taking place, the critical parameter is the number of parallel calls (PC). Our lab measurements in this configuration have shown the following capacity for the following instance types available on the Amazon Marketplace: (the instance parameters are from https://aws.amazon.com/ec2/instance-types/)

Instance Type	PC	vCPU	Mem (GiB)	Networking Performance	Notes
m3.medium	180	1	3.75	Moderate	CPU-constrained
m4.large	372	2	8	Moderate	network-constrained

In the less usual case that SIP is processed without RTP, number of call attempts becomes the critical parameters. This can be the case when the SBC is used as a signaling-only load-balancer. Then choosing a CPU-strong instance type makes sense. Our tests have shown that the m3.xlarge instance type can deliver 40 Calls Per Second signaling rate, c3.8xlarge delivers about 500 CPS.

Note that OS-reported CPU-load values may be misleading on virtualized machines. CPU time may be "stolen" by virtualization hypervisor and system tools may or may not accurately report the status. The more accurate method to determine actual utilization of the virtual instances is CloudWatch. We recommend that CloudWatch-observed CPU utlization shall not exceed 80% – if deployed in an Elastic Cluster, this should be the threshold value triggering autoscaling cluster growth.

6.16 Template parameters

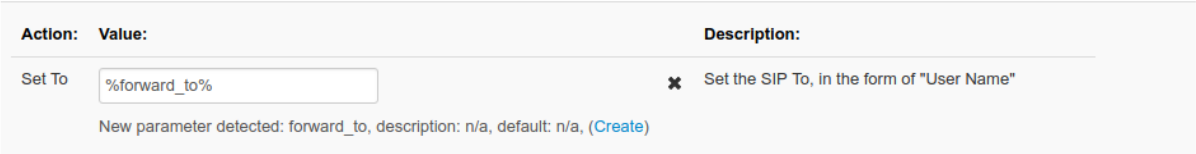
When configuring multiple SBC nodes, it might be needed to use different value in a rule or config variable for certain nodes or config groups. Template parameters can be referenced in various configuration parameters and thus allow for building different values for specific nodes or complete config groups. Template parameters are referenced by their names enclosed by a percent characters ("%"). For example: **%my_param%**. Each instance of a template parameter is replaced by its value when the configuration for a specific node is generated.

6.16.1 Definition of Template Parameter

There two ways to define new template parameter:

Define param directly in input field

Just enter its name enclosed by "%" characters into any input field allowing template parameters and click on the 'Create' link.

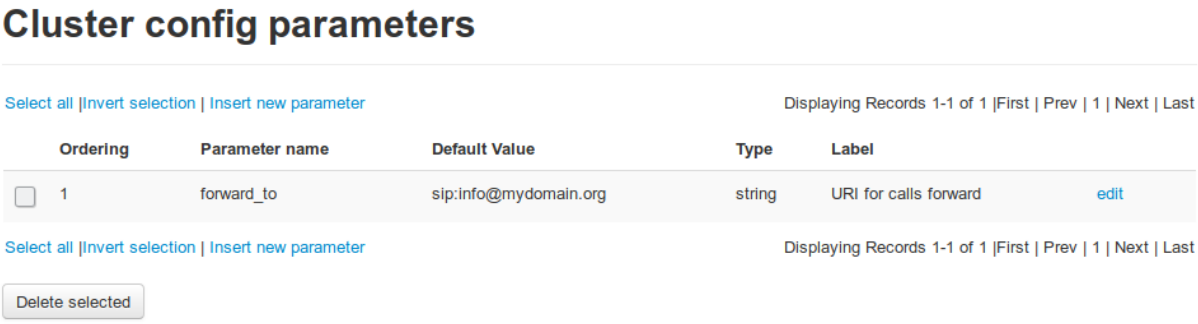


If a template parameter needs to be used in a drop-down or checkbox field in the GUI, just simply click on the pencil icon next to the input field. It allows for entering free form text into the input field.

As of now the template parameters are supported in *Rules*, *Global Config* and in interfaces/node TLS profile assignment.

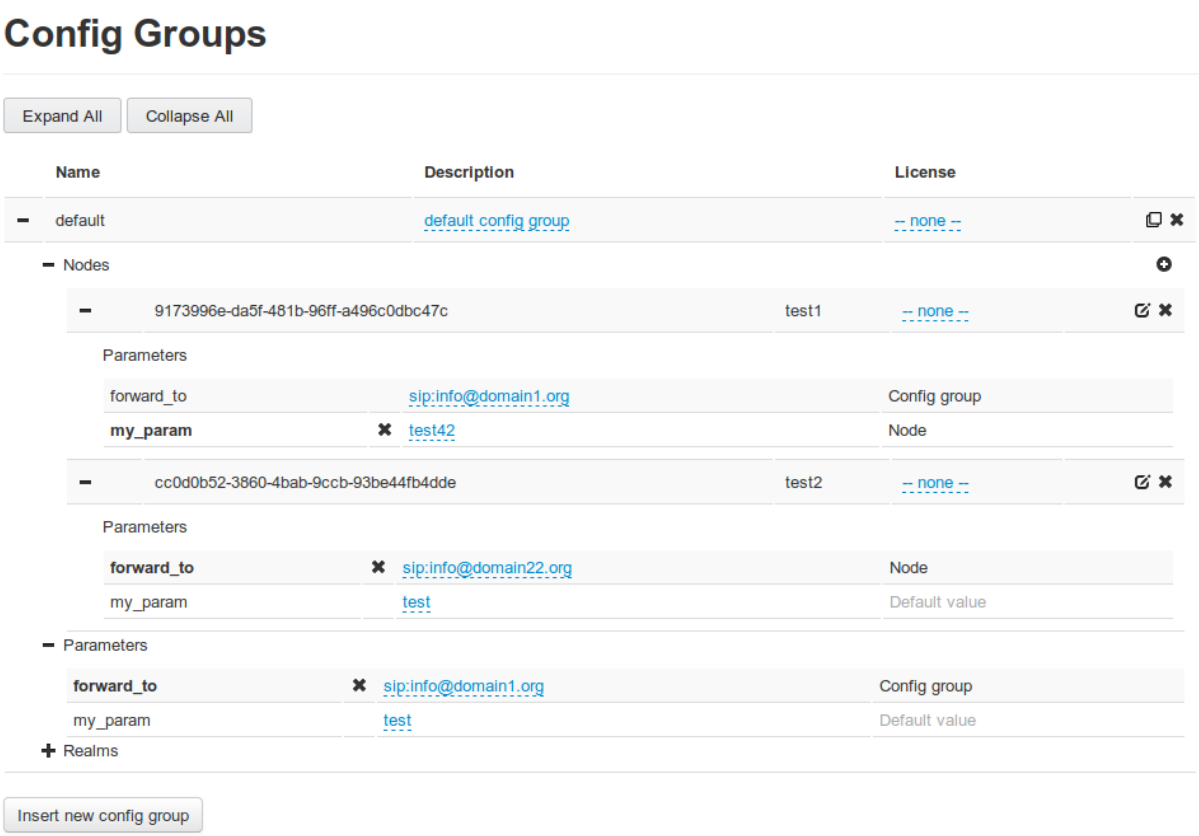
Define parameters on the "Cluster config parameters" screen

The screen located at "Config->Define cluster config parameters" lists all the defined parameters. The screen allows for creating new parameters as well.



6.16.2 Set specific values for Template Parameters

Once the parameters are defined, their value can be customized per node or config group on the "System -> Config Groups" screen.



If the value for a defined parameter is not customized, its default value is used. Otherwise it is replaced by the config group value or node value (node specific value takes precedence over config group value).

Chapter 7

ABC SBC System administration

This Section describes the administrative tool available on the ABC SBC. There is also the CLI reference, see *Command Line Reference*.

7.1 User Management

There are two ways how to administer User Accounts and granular access control for the ABC SBC: using GUI and using CLI. The primary method is via GUI, the CLI method is restricted in functionality. Both methods are described in the following subsections.

In opposite of other SBC configuration the changes in user accounts take immediate effect. They do not require activation of SBC configuration.

The access control concept is based on the notion of group membership. Groups define at a granular level permissions to perform specific actions, such as GUI access, viewing and/or modifying the ABC SBC topology, monitoring various aspects of the ABC SBC, accessing RPC, firewall administration, etc.

A user gains all associated privileges by being assigned to a group. A user can be member of multiple groups.

The system comes with preconfigured user accounts as described in the Section *Default User Accounts*.

7.1.1 GUI User Management

Access to the administrative GUI can be managed using User Management (menu: *"System → Users"*) and Group Management (menu: *"System → Groups"*).

User Management allows to add, delete and modify users – their passwords and group membership.

Group Management allows to enable/disable the respective permissions for a group. Once set and applied, it applies to all group members.

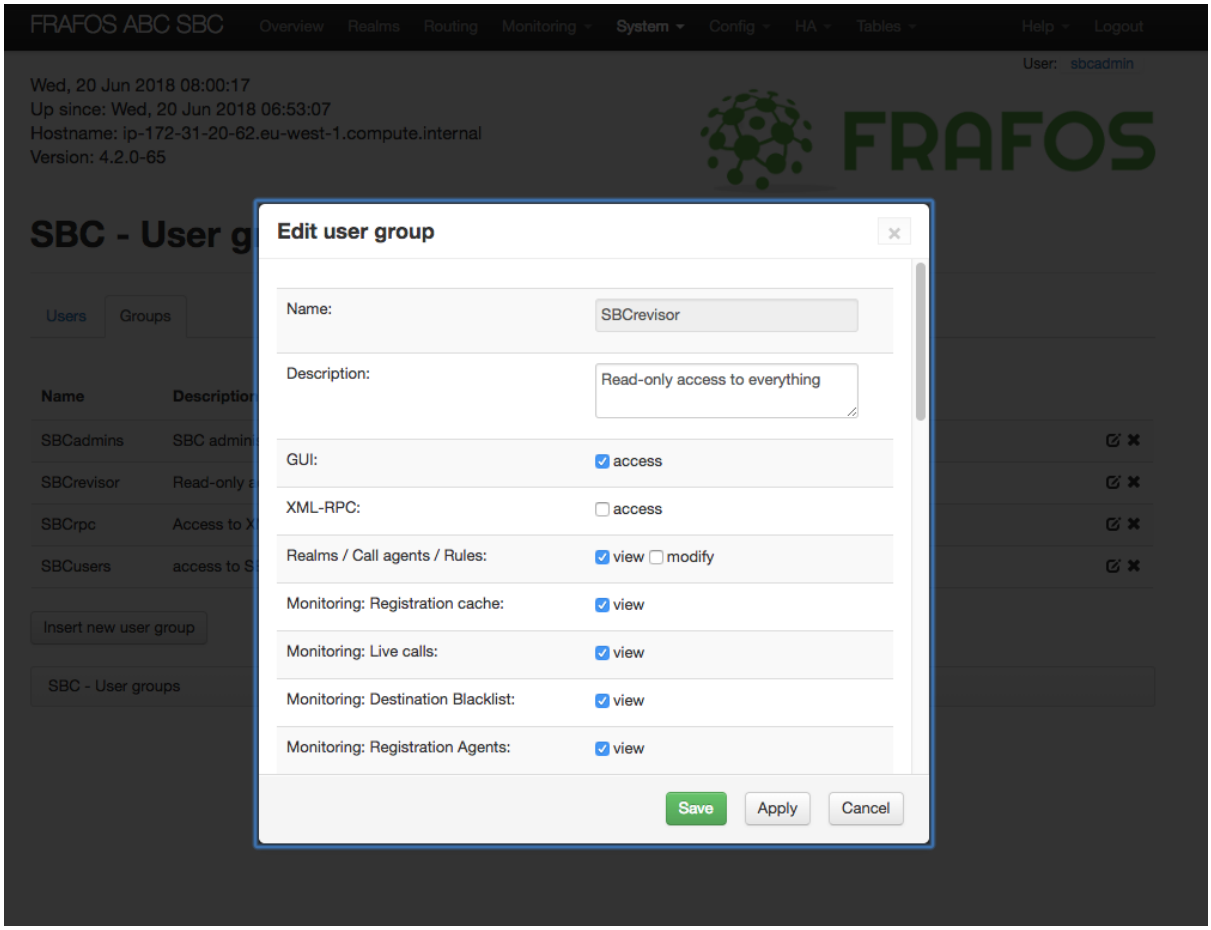


Fig. 1: Example: Group Management

7.1.2 CLI User Management

The CLI User management permits to create new users and assign them to a group. This subsection lists available commands.

To add a new user, use the CLI the following way:

```
% sbc-add-user '--password=DoyoulikeH.323?' admin
```

The new user comes without any privileges and must be assigned to a group. To assign the new user to the administrative group with all permissions use the following command:

```
% sbc-add-user admin SBCadmins
```

Additional commands include:

- **sbc-del-user** to delete a user
- **sbc-list-groups** to show existing user groups
- **sbc-list-users** to show existing users
- **sbc-user-passwd** to change a user's password

7.2 Server Administration

If maintenance of the server running ABC SBC is needed, it is possible to restart the server from the GUI. There are the following buttons on Server Administration screen, accessible using the "System → Administration" link:

- Shutdown: performs soft shutdown of the server.
- Restart: performs soft restart of the server.
- Halt: performs halt of the server (meaning the same as shutdown, but with no power off at the end).
- Force HA offline: displayed only if node is running as HA pair, it puts forcibly the node into HA FAULT mode. In this mode the HA resources (VIP addresses, routes) and signalling application are stopped on the node, and should be moved to the other node which becomes new HA MASTER (under the condition that the other node is up.) It can be used when upgrading ABC SBC, to make sure the node being upgraded is not processing traffic. The same can be achieved also using from command line using "sbc-ha-offline" command on the specific node.
- Un-force HA offline: displayed only if node is running as HA pair, reverts the forcibly set HA FAULT mode that was set using "Force HA offline", meaning it puts the node back to normal mode, in which a node can be either HA BACKUP or MASTER depending on negotiation with the other node. The same can be achieved also using command line command "sbc-ha-online" on the specific node.
- Current HA status can be checked using "sbc-ha-status" command line command on the specific node.

7.3 Backup and Restore Operations

7.3.1 ABC SBC Configuration Management

The ABC SBC configuration is stored in a local MariaDB database on the configuration master node. When the administrator applies the changes using the **"Activate SBC configuration"** link, an automatic snapshot of the configuration database is created and is labeled as „Automatic Snapshot" in the list of available snapshots.

Further, when an upgrade of the system is triggered, see Sec. *Upgrade Procedure*, a snapshot of the configuration is created before the upgrade and is labeled as „Upgrade Snapshot" in the list of available snapshots

The SBC administrator can manually trigger the generation of the configuration DB snapshot from the GUI. When the configuration snapshot is created, it is recommended to write a short comment to note what exactly has been modified. Optionally also content of provisioned tables database can be included in the snapshot.

These configuration DB snapshots can be accessed using the **"System → Config Management"** screen, see Fig., *Managing the ABC SBC configuration backups*. From the GUI, the administrator can create new snapshots or change or add a comment to an already existing snapshot. To restore a saved configuration the administrator can use the **"Load config"** link of the desired configuration snapshot, or the **"Load provtables"** link to load the content of provisioned tables (if the snapshot contains it).

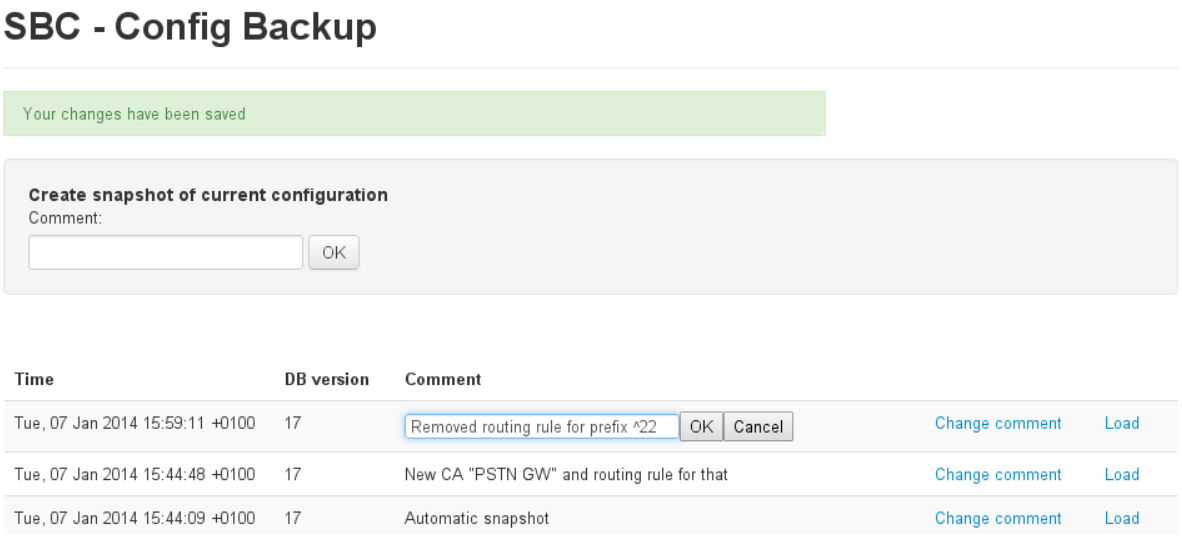


Fig. 2: Managing the ABC SBC configuration backups

Note: When the configuration DB backup is loaded, the configuration is NOT automatically applied. The administrator should check if the restored configuration is the correct one and then has to manually apply it using the "Activate SBC configuration" link.

7.3.2 ABC SBC Configuration Backup

Apart from the above configuration snapshots, it is also possible and recommended to use automatic daily ABC SBC backups, which can be enabled under Config / Global Config / Backup tab. The following options can be set there:

- **Create daily Sbc configuration backups** - this enables the daily backup.
- **Include provisioned tables in daily backups** - when enabled, also content of whole provisioned tables database will be included in the daily backup. It is enabled by default and recommended.
- **Number of days to keep backups** - sets the retention period for the daily backups. On each backup run, all backup files older than the specified number of days will be deleted. Use 0 to disable any automatic removal.
- **Destination directory for backups** - sets the directory, to which the daily backup files will be created. Default setting is "/data/backups". The partition holding this directory should have enough space for the daily backups.
- **Include sensitive system data (ssh keys, passwords) in backup** - if enabled, also system ssh daemon setting, system keys, root user authorized keys, and system password files (passwd, shadow) will be included in the backup.
- **Run daily backups on config master node only** - By default, the daily backup is performed on all ABC SBC nodes. On the nodes that are not configuration master, the backup does not include the main configuration, but still can be used for easier node recovery. If this option is enabled, the daily backup will be performed only on node set to act as main configuration master.

It is highly recommended to enable the daily backups and include the backups destination directory to customer off-server backups to external device, or at least to copy the backup files to external device after important changes done on ABC SBC configuration, to be able to recover SBC node in case of hardware failure. Note that to minimize external backup impact on ABC SBC performance, a solution allowing to use only idle I/O and CPU should be used.

The daily backup files are gzipped tarball archives and contain the following data, which can be used (directly or as a reference) when recovering ABC SBC configuration: the main ABC SBC configuration database dump (backups from master node), optionally also whole provisioned tables database dump, versions of important rpm packages installed, local configuration files templates (if existing), system network interfaces configuration files, system hostname, system hosts file, MariaDB server configuration file, node uuid info, optionally also root user ssh authorized keys system password files.

7.3.3 ABC SBC Recovery Procedure

In case ABC SBC server dies, it is possible to recover it using the following steps. In case more ABC SBC nodes are used, this procedure differs depending on if recovering main configuration master node or not.

Steps to be done when recovering configuration master node (or single node setup, where the node is configuration master for itself):

- Locate latest ABC SBC backup file from the daily ABC SBC backups.
- Install clean CentOS OS on the new server.
- Configure system network interfaces, routes, hostname. If needed, the system configuration files included in backup can be used as a reference.
- Install ABC SBC on the server, following normal installation steps, see Sec. *Installation Procedure*. Use the same major release line (like 4.0.x or 4.1.x) that was there before. Note: it will install latest version of ABC SBC from the used major release line. If for any reason restore to exact older version of ABC SBC is needed, contact Frafos support at this point.
- Perform only the "sbc-set-master" and "sbc-init-config" initial configuration steps.

Important: when performing the initial configuration, it is highly recommended to provide the node uuid that was used previously on the node being recovered. It can be found in file named "node_uuid" in the backup directory or inside the backup tarball file. If the previous node uuid is not provided, the node can be still recovered, but in case there was some configuration specific for that node (like system interfaces assigned to that particular node), it will have to be fixed to apply to newly created node in GUI.

- Restore ABC SBC configuration from the backup, by calling the following command, where the <backup-file> is the daily backup gzipped tarball file. Optionally use also "–prov" option to restore also provisioned tables content, if it was included in the backup.

```
% sbc-restore --rest-ver --bckfile <backupfile>
```

- Access ABC SBC administration GUI and check the restored configuration. In case system interfaces names differ on the new server, update the logical to system interfaces mapping under System / Interfaces.
- In specific case when the configuration restored on config master was not the latest one and the SBC slave nodes already use configuration with higher "configuration version number", it is needed to force it to be higher on the config master side. The configuration version number that SBC nodes use can be seen on "Nodes status" page per each node, it is shown in red color if differs from the latest exported configuration from config master. The following command can be used on config master node to forcibly set the exported configuration version number to specific value:

```
% sbc-set-confversion <version number>
```

- Activate the configuration using "Activate Sbc configuration" link, which is available at bottom of Overview GUI page.
- Check if the ABC SBC pulled new configuration using Monitoring / Nodes status.

Steps to be done when recovering a node not being configuration master, where the configuration master node is still working:

- Install clean CentOS OS on the new server.
- Configure system network interfaces, routes, hostname.

- Install ABC SBC on the server, following normal installation steps, see Sec. *Installation Procedure*. Use the same major release line (like 4.0.x or 4.1.x) that was there before. Note: it will install latest version of ABC SBC from the used major release line. If for any reason restore to exact older version of ABC SBC is needed, contact Frafos support at this point.
- Perform only the "sbc-set-master" (do not enable config master on the node being recovered) and "sbc-init-config" initial configuration steps, provide existing configuration master node address.

Important: when performing the initial configuration, it is highly recommended to provide the node uuid that was used previously on the node being recovered. It can be found under "Details" of the node on "Nodes status" page or on "Nodes" page under "System" menu on ABC SBC config master GUI. If the previous node uuid is not provided, the node can be still recovered, but in case there was some configuration specific for that node (like system interfaces assigned to that particular node), it will have to be fixed to apply to newly created node in GUI.

- Access ABC SBC administration GUI on configuration master node and check configuration. In case system interfaces names differ on the new server, update the logical to system interfaces mapping under System / Interfaces.
- Activate the configuration using "Activate Sbc configuration" link on configuration master node, which is available at bottom of Overview GUI page.
- Check if the recovered node pulled new configuration using Monitoring / Nodes status on configuration master node GUI.

7.3.4 Manual Backup of the Complete SBC Configuration

It is also possible to manually backup the important files:

- **SBC database: administrator can manually create a full SBC** configuration backup using **``sbc-backup``** command which creates a backup files into *``/data/sbc/configs``* directory, where also automatic DB backups are stored.

The following command line options can be used:

```
% --prov
```

Optionally also content of provisioned tables db can be included.

```
%  --comment <comment>
```

A backup comment can be specified.

```
%  --bckfile
```

If used, the backup files will be put also to gzipped tarball. Filename will be automatically generated from date, version and comment.

```
%  --bckdir <dir>
```

Specifies the directory where to save the backup file, if –bckfile option is used. Defaults to /data/backups.

```
%  --filename <file>
```

If used together with –bckfile, save the backup under specified full file pathname instead of automatically generated name.

```
%  --remove
```

If the –bckfile option is used, by default the backup files are left also in the default directory (/data/sbc/configs/). When this option is used, they will be deleted from that directory after creating the gzipped tarball file.

```
%  --incl-ssh
```

If used, system ssh daemon settings and root keys and authorized keys will be included in the backup too.

```
% --incl-extra
```

If used, extra custom files will be added to the backup. The extra files can be listed using Global Config setting under Config / Global Config / Backup tab, using full paths to the files, with separating more fields by comma. There is a limitation that the filename cannot contain comma character.

```
% --incl-all
```

Enables inclusion of all provisioned tables, system ssh settings and keys and password files at once.

```
% --excl-tls
```

Do not backup TLS profiles. This option can be used only when creating backup of config master node.

```
% --incl-dbsnaps
```

Include also configuration database snapshots. This can be used only on config master node.

```
% --quiet
```

Print only warnings and errors, no info messages.

- **CDRs:** the customer's billing system should regularly download CDRs generated by the SBC which are stored on the SBC for 93 days by default. CDRs are stored to the *"/var/log/frafos/cdr"* directory.
- **Logs:** log files are stored in *"/var/log/frafos"* directory
- **Traffic logs:** traffic logs created by the **"Log received traffic"** action are stored in the *"/data/traffic_log"* location.
- **Recording files:** recording files are stored in the *"/data/recordings"* location.

7.3.5 Manual Restore of the Complete SBC Configuration

For manual backup restore, the command "sbc-restore" can be used. Be careful when using it, as it can overwrite also various system files. The following options can be used:

```
% --bckdir <dir>
```

The backup from specified directory will be restored.

```
% --bckfile <filename>
```

The backup from specified backup gzipped tarball file will be restored.

```
% --prov
```

Restore also provisioned tables, if those are included in the backup.

```
% --provonly
```

Restore only provisioned tables, do not restore the main SBC configuration.

```
% --rest-ver
```

Normally, when the SBC configuration is restored, the "configuration version number", which is used by nodes to detect that newer configuration is available from config master node, is not restored but the current number kept. That allows administrator to review the restored configuration in GUI and then activate it, making SBC nodes pull new configuration. This options forces that also the configuration version number is restored to the value from backup, which is needed when doing full config master node recovery after clean reinstall.

```
% --rest-ssh
```

Restore system ssh settings, keys and root user authorized key files, if present in the backup. Use with caution.

```
% --rest-net
```

Restore system network configuration files (/etc/sysconfig/network-scripts/*). Use with caution, make sure that the system network interfaces are using still the same names if recovering server. Note that it just restores the files and either system reboot or network service restart may be needed to aplly the new erstored settings.

```
% --rest-sysfiles
```

Restore system files /etc/hostname and /etc/hosts and mysql config my.cnf (if running on CCM). Note that it just restores the files, does not change current hostname.

```
% --rest-system
```

Restore all system files (ssh, network, hostname etc). Use with caution.

```
% --rest-sbc
```

Restore all Sbc files (config version, provisioned tables, config database snapshots etc).

```
% --rest-sbcpull
```

Restore Sbc config pull and related config files, including node uuid and local config templates. The Sbc config pull configuration is the info that was initially entered using sbc-init-config command. If client certificate is used for config pull, it is restored too.

```
% --rest-dbsnaps
```

Restore config database snapshots, if included in the backup tarball. Can be used only on config master and only when restoring from tarball.

```
% --rest-extra
```

Restore extra custom files, if included in the backup. Note that while restoring the extra files, the file permissions and ownership are preserved, but in case the directories for the files are missing and have to be re-created while restoring, the directories permissions and ownership are not preserved.

```
% --rest-all
```

Restore everything included in backup (Sbc config, provisioned tables, ssh, network, system, sbc pull config).

```
% --excl-tls
```

Do not restore TLS profiles, if those are included in the backup.

```
% --quiet
```

Print only warnings and errors, do not print info messages.

```
% --rootfs <path>
```

This is lowlevel option allowing to run the sbc-restore command e.g. on virtual machine or container instance which is stopped and the Sbc filesystem is mounted to some directory of the host system. The provided path should point to the directory where the Sbc filesystem is mounted. The restore operations will just overwrite files but skip all steps that would require running system.

7.4 ABC Monitor Backup and Restore Operations

7.4.1 ABC Monitor Configuration Backup

A backup of the ABC Monitor main configuration file can be created from commandline using "abc-monitor-backup-config" command.

By default, the file name of the backup is created automatically using current date and time and the file is saved to "/data/backups/configs" directory.

The configuration backup contains only ABC Monitor application settings, it does not contain any system settings (like hostname, network interfaces settings and similar).

The following options can be used:

```
%  --file <filename>
```

Specify full path of the file to which the configuration backup should be saved. If not used, the file name is created automatically and default directory used.

```
%  --quiet
```

Do not write any info messages. If used, only errors are written, plus a message like the following: "Config backup saved to: /data/backups/configs/monitor-2020-01-16_14-54-18.json".

7.4.2 ABC Monitor Configuration Restore

A backup of the ABC Monitor main configuration can be restored from commandline using "abc-monitor-restore-config" command. It checks the configuration file, restores it as the main config file and optionally also activates the new configuration.

The following options can be used:

```
%  --file <filename>
```

Sets full path to the file with the ABC Monitor configuration backup to be restored. This option is mandatory.

```
%  --noactiv
```

By default, the restored ABC Monitor config is activated, meaning that services which are affected by the new config are restarted. If this option is used, the config backup is restored but no activate done. Admin can review the restored config in ABC Monitor gui and use Save from there to activate it.

```
%  --quiet
```

Do not write any info messages. If used, only errors are written.

7.5 Howto setup a Semi-redundant CCM on ABC SBC

This section describes steps needed to setup a "semi-redundant" cluster config master (CCM) node for Frafos ABC SBC.

With current ABC SBC release, there is no official support for redundant CCM node. But with help of this document, a backup CCM node plus automatic transfer of configuration backup to it can be set up, which will be ready to take over the role of cluster config master for SBC nodes in case of main CCM node failure.

Note: the official full support for geo-redundant CCM is planned for future ABC SBC release.

We refer here to the main active CCM node as "primary CCM" and to the backup node as "backup CCM".

The procedure is based on standard ABC SBC backup / restore using configuration snaphots, as described in ABC SBC handbook Sec. *Backup and Restore Operations*. Please refer to that for details.

These steps assume both SBC and CCM nodes deployment via systemd based containers, but the procedure can be used on standard SBC installation as well, the ssh port numbers need to be updated according to particular customer setup configuration.

Note that the switch to backup CCM still requires manual intervention.

7.5.1 Setup primary CCM node

Start and configure the primary CCM node the standard way.

After initial configuration is done, note the primary CCM node "node uuid" value, which can be found on GUI "System" / "Nodes" screen, value from the row for the CCM node itself.

7.5.2 Setup backup CCM node

Start clean fresh CCM container (or standard installation) for the backup CCM (on some other physical host), using the same SBC release and the same installation way as the primary CCM, but do not configure anything in GUI. If configuring non-container CCM, perform also the "sbc-set-master" step according to standard initial configuration procedure.

Access the backup CCM node command line as root user (either via ssh or using the container console) and perform the following command:

```
% sbc-init-config
```

For the values prompted just press Enter to use default values, or enter the same values as on primary CCM, if non-default values are being used. On the "In case specific node uuid is required, please enter it" prompt, paste the primary CCM node uuid, as noted in the previous step done on primary CCM node.

7.5.3 Configure configuration snapshot backups

On primary CCM GUI, navigate to "Config" / "Global config" screen, and there select the "Backup" tab.

Enable the "Create daily Sbc configuration backups" option.

Make sure that also "Include provisioned tables in daily backups" option is enabled.

Enable also "Run daily backups on config master node only" option.

Optionally enable also "Include sensitive system data (ssh keys) in backup", if restoring e.g. ssh keys and hostname on backup CCM node is required.

The option "Destination directory for backups" sets directory, to which the daily configuration snapshots are saved. Default directory is "/data/backups" local directory on the primary CCM node. It can be also possibly pointed to a specific directory which is mounted externally to the primary CCM node, e.g. from external network share device or via NFS.

Activate new configuration from CCM GUI.

Optional step: in case the backup done daily would be too low frequency, it is possible to change that according to customer need, e.g. to be done every hour. The command which performs the daily backup is "sbc-daily-backup" and by default it gets started from "/etc/cron.daily/sbc-backup". This can be customized by administrator e.g. by moving the file to cron daily directory, to make it run every hour:

```
% mv /etc/cron.daily/sbc-backup /etc/cron.hourly/
```

But please consider the fact that the configuration snapshots contain also all provisioned tables data, so they can be big. Also the change like this, to set more frequent backups, won't survive possible CCM container replacement with newer one.

7.5.4 Setup configuration backups transfer to backup CCM node

The configuration snapshot backup files need to be transferred from the primary CCM node to some other safe location, to ensure they do not get lost in case of primary CCM node failure, or can be transferred directly to the backup CCM node.

The recommended way is to manage the files transfer from other external customer server, not from the primary CCM node itself, to avoid loosing that functionality when e.g CCM container is replaced with newer one.

Depending on customer deployment, possible ways to achieve this are like:

- If the daily backups on primary CCM node are created to some externally mounted directory, customer can setup some regular way of copying those files to the backup CCM node "/data/backups" directory, e.g. using "scp" secure shell copy, or "rsync" program. In this case please also pay attention to available space on target location on backup CCM and possibly delete old files (rsync option "`--delete`" can be used for that).
- Another option is that the latest configuration backup can be copied from external location to the backup CCM only when primary CCM failure happens.
- If the configuration snapshot backups are created only to "/data/backups" local directory on primary CCM, it is possible to setup e.g. "rsync" command to be run periodically, which will transfer whole "/data/backups" directory content in a efficient way from primary CCM node to backup CCM node directly, using ssh as underlying protocol. The transfer can be set up to be initiated either from primary CCM side or from backup CCM side. We recommend to initiate it from backup CCM side.

Example of rsync command to synchronize the configuration backups (assumes ssh on port 24 on primary CCM node, 192.168.0.1 is the IP addr of the primary CCM), to be run on backup CCM node:

```
% rsync --delete -r -e 'ssh -p 24' root@192.168.0.1:/data/backups /data/
```

This command can be run periodically e.g. using system "cron" service, by creating a file like "/etc/cron.d/rsync" with the following content on the backup CCM node, to make it run every hour (at 10 minutes past every hour):

```
10 * * * * root rsync --delete -r -e 'ssh -p 24' root@192.168.0.1:/data/backups /
↪data/
```

Note: if ssh is being used as underlying protocol for rsync, it is possible to make it work from backup CCM to primary CCM without passphrase using the following commands on the backup CCM (192.168.0.1 is the IP address of primary CCM):

```
% ssh-keygen
% ssh-copy-id -p 24 root@192.168.0.1
```

After the setup of configuration backups transfer is done, make sure that the backup files are really being transferred automatically to "/data/backups" directory on the backup CCM node.

Check also the backup size and available space, and tune global config setting "Number of days to keep backups" on primary CCM GUI (Backups tab). Note that if using the rsync command (with the "`--delete`" option), the files deleted on primary CCM node directory will be also deleted automatically by rsync from the backup CCM node directory.

7.5.5 Steps to make the backup CCM available in case of primary CCM node failure

In case of primary CCM node failure perform the following steps:

- Find the latest configuration backup file that was transferred to backup CCM directory "/data/backups", or if using external backup location copy it to backup CCM "/data/backups" directory.
- Restore the backup using command like this on the backup CCM:

```
% sbc-restore --prov --rest-ver --bckfile
  /data/backups/sbc-backup-2020-10-22_11-36-50_42001027_4.2.22-77_daily_backup.tar.
↪gz
```

Note: if system stuff like ssh keys or hostname was included in the backup too, and restore of that is needed, add also the following option:

```
--rest-system
```

- Access the backup CCM GUI and review the loaded configuration. Pay special attention to "System" / "Interfaces" records for the CCM node itself. If the backup CCM IP address used on IMI or XMI interfaces is different from IP address used previously on primary CCM, update it. (Not needed if type of IP is set to "Autodetect")
- Activate the new configuration from backup CCM GUI, to make it available for SBC nodes.

7.5.6 Steps to be done on SBC nodes to start using new CCM

Once the backup CCM is available and configuration snaphsot backup was loaded on it, and if the backup CCM uses different IP address from the previous main CCM, re-configure all SBC nodes to use new CCM address using the following command on each of them:

```
% sbc-init-config
```

Alternatively, a DNS hostname can be used as CCM node address on all SBC nodes. In that case it is recommended to use a DNS record with short TTL value, which allows then easy central change of the CCM address just by updating the DNS record, without need to update it on all SBC nodes. (Note: but avoid using more A records under one DNS name, pointing to more IP addresses).

7.5.7 Additional steps and checks

Access the backup CCM GUI "Monitoring" / "Nodes status" screen. Check if all SBC nodes have pulled new configuration from the new CCM.

There may be a duplicate "Nodes status" record shown for the CCM node itself (coming from the node uuid update done initially), but this older CCM node status (which can be identified by the "Last report" column) can be safely ignored, or deleted if using newer CCM which allows that.

Note: in specific case, when the configuration snapshot that was restored on backup CCM was not the latest one, and if the "sbc-init-config" step was not done on Sbc nodes, the nodes will not pull the configuration from the backup CCM after switch to it automatically, because the "configuration version" number used to detect new configuration will be lower on the backup CCM than what the nodes already expect. This should not happen if following this procedure correctly and latest configuration snapshot was restored on the backup CCM. But in case it happens, which can be seen on backup CCM GUI screen "Monitoring" / "Nodes status" by the nodes configuration versions higher than the "Latest config version", it is possible to manually forcibly increase the configuration version of configurations exported from CCM using "sbc-set-confversion <version>" command.

7.6 Upgrade Procedure

FRAFOS regularly releases a new version of ABC SBC. New features, modifications and bug fixes are described in a "Release notes" section of SBC handbook for every new release.

Even if the upgrade should not cause a service disruption, it is recommended to perform this procedure in the service maintenance window.

If HA is used, before the upgrade is started, both cluster nodes have to be online and all required services running, see Sec. *Command-line SBC Process Management* for more details. The administrator should also create a configuration snapshot, see Sec. *Backup and Restore Operations*.

If separate CCM and SBC nodes are used, please upgrade the CCM node first, and continue with SBC node(s) upgrade only after new CCM is verified to be correctly functional.

The following upgrade procedure applies to standard ABC SBC installation. For container installation, see following *Container ABC SBC upgrade* section and for ABC Monitor upgrade see *ABC Monitor Upgrade Procedure* section.

7.6.1 Standard ABC SBC upgrade

To upgrade to new ABC SBC version, run the following command:

```
% sbc-upgrade
```

The ABC SBC configuration database backup is performed first, then new RPM packages are downloaded and upgraded automatically.

Important: In case when any package is not successfully updated, contact FRAFOS support.

It is also recommended to reboot the machine to be sure that all updates (e.g. kernel update) are correctly applied and loaded. The administrator should then check the server status, see Sec. *Command-line SBC Process Management*.

7.6.2 Container ABC SBC upgrade

When the ABC SBC is deployed as a container, there is no "online upgrade" of the existing (and running) container, but the whole container is replaced by newer version.

It is highly recommended to use separate directory "mounted" to the container for "/data" path, as described in the container install section, which keeps data that is expected to be persistent and makes the container replacement easier. If it is not used, it is possible to manually copy or move the /data sub-directories of old container after stopping it to new container before starting it for the first time. If doing so, please pay attention to keeping files permissions and ownership.

When replacing container, please follow these steps:

- Create a ABC SBC backup. Note: the backup file is needed when replacing CCM node container, but might be needed also in case of troubleshooting possible issues, so create it on both CCM and SBC nodes. Use command like this on container:

```
% sbc-backup --incl-all --bckfile
```

It will create backup under "/data/backups/" directory by default. Note the created backup tarball filename.

- Stop the container.
- Backup directory with the old container, by renaming the directory like:

```
% cd /var/lib/machines
% mv <name> <name_backup>`
```

- Create new directory (using the same name as before) and unpack the new container image to it, similar way like listed also in the install section - example:

```
% mkdir /var/lib/machines/<name>
% tar --xattrs -p --numeric-owner -C /var/lib/machines/<name> \
% -xzf frafos-abc-sbc-4-6-1-481.tgz
```

- If the externally mounted "/data" persistent directory is not used, as mentioned above, copy or move content of old container "/data" sub-directory to new container "/data" sub-directory, example:

```
% cp -a /var/lib/machines/<name_backup>/data/* /var/lib/machines/<name>/data/
```

- Start the new container.
- Login to the new container console.

If the container is CCM (cluster config master):

- Call command to enable CCM:

```
% sbc-set-master --enable
```

- If persistent /data directory is not used, call the following command to perform the initial config:

```
% sbc-init-config
```

- Restore the configuration backup, pass the filename of backup created in first step before, example:

```
% sbc-restore --rest-sbc --bckfile <path to backup file>
```

- Access the CCM GUI, review the restored configuration and activate it. Alternatively, activate new configuration using the following command:

```
% sbc-publish-config
```

If the container is SBC node:

- Do not restore the config backup, unless there is some specific reason for doing that.
- If persistent /data directory is not used, call the following command to perform the initial config:

```
% sbc-init-config
```

- The SBC should automatically pull configuration from the CCM node.

Access the CCM node GUI and check Monitoring / Nodes status page, check for any errors reported by SBC nodes.

7.7 ABC Monitor Upgrade Procedure

To upgrade to new ABC Monitor version, run the following command:

```
% abc-monitor-upgrade
```

The ABC Monitor new RPM packages are downloaded and upgraded automatically.

Important: In the case when any package is not successfully updated, contact FRAFOS support.

If the ABC Monitor is deployed as container, and the container replacement using newer version is to be performed, the new ABC Monitor can be either configured again via GUI, or the configuration can be transferred from old container to new container. Please refer to *ABC Monitor Backup and Restore Operations* section for details about the configuration backup and restore.

The old events data can survive container replacement with newer version, if the "/data" directory of the container is either mounted externally (and not erased while setting up the new container), or if the "/data" directory content is copied or moved (while keeping ownership and access rights) from old stopped container to new unpacked container, before starting it for the first time.

Note: the "mounting" of persistent host directory to the "/data" directory of the container can be achieved using `"--bind"` option of "systemd-nspawn" command used to start the container.

Note: older ABC Monitor releases up to 4.4 used different location "/var/lib/elasticsearch" for storing the events data, while starting with release 4.5 the directory was moved to "/data/elasticsearch" to allow possible persistence across container replacement, if the whole "/data" directory is mounted externally to the container.

Note 2: If data format has changed, some charts or Monitor function maybe will not work correctly with old data format.

7.8 SBC Dimensioning and Performance Tuning

This section provides background information on typical traffic patterns, its performance implications, and performance tuning possibilities. This information can help to make a more educated estimate than provided in Section *Capacity planning*. However, confidence can only be achieved by measurement of the target ABC SBC configuration against actual traffic on the used hardware.

The reference hardware we used is Sun SunFire X4170 with the following configuration:

- 2 x Intel Xeon X5570 @ 2.93GHz CPUs, each 4 cores with hyper-threading enabled
- 2 x on-board Intel Gigabit ethernet adapter
- 8 GB RAM

As an alternative, we measured on a Dell R410 with the following configuration:

- 2 x 4-core Intel X5550 CPU 2.6GHz
- 2 x Broadcom NetXtreme II BCM5716, 8 irqs/queues
- 12 GB RAM

The alternative results are shown in parenthesis.

On the reference hardware, the maximum performance limits of the ABC SBC have been measured as follows:

- 5000 parallel G.711 calls with media anchoring (3600)
- down by factor of five when transcoding is used,
- call rate of 480 calls per second, without media anchoring
- registration rate of 9900 registrations per second.

Actual use-cases may have significantly different traffic characteristics and therefore the resulting performance may be driven by different limits. In this section we look at the most typical cases: a trunking case with and without transcoding and a residential deployment scenario. All the use-cases assume a single-pass SBC traversal for both SIP and RTP. In the next section, we also summarise the critical configuration aspects that need to be checked when tuning the system for the highest performance.

7.8.1 Trunking Use Case

This case is characterised by handling many calls, both signalling and media, from rather few sources. SIP traffic is not NATed and does not include REGISTER transactions. In this case, the most demanded functionality is media forwarding and the most significant bottleneck is the packet rate of the Ethernet card. Small packets as common with VoIP saturate Ethernet card nominal capacity much earlier than large HTTP packets would. A reasonable Ethernet card shall deliver at least 400 thousand packets per second in each TX and RX direction.

With such a packet rate, the following number of parallel calls can be achieved for the respective number of calls:

codec/packetisation	number of calls
G.711/20ms	5,000 (3,600)
G.729	6,000 (4,680)

7.8.2 Trunking with Transcoding

Transcoding is typically deployed as an additional feature in the trunking case. However, as transcoding is computationally expensive the bottleneck shifts to CPU. The following numbers are achievable on the reference platform:

transcoding	number of calls
G.711-to-G.729	1,000 (750)

7.8.3 Traffic Estimates for Residential VoIP

With residential VoIP, the deployment sees many challenges: the clients are connecting from unmanaged networks over NATs and variety of SIP client types causes interoperability issues. Addressing NAT traversal by enforcing media anchoring (see Section *Media Anchoring (RTP Relay)*) and frequent re-registration (Section *Registration Handling Configuration Options*) causes substantial increase in overhead. The ABC SBC keeps the heavy SIP traffic off the infrastructure behind it, however the bandwidth impact on the incoming side must be considered.

Without frequent re-registrations NAT address bindings would expire and SIP devices behind NATs would loose incoming traffic. While other more light-weight methods (STUN, CRLF) exist, re-registrations are safe in that they work with every SIP client and create traffic keeping any NAT bindings alive. The penalty is quite high resource consumption. The "background SIP traffic" is even higher in public SIP services than one could infer from baseline calculation based on re-registration period. Alone use of digest authentication doubles number of REGISTER transactions, many clients send additional traffic to check voicemail status (SUBSCRIBE), announce their online status (PUBLISH), and get over NATs on their own (OPTIONS). As a result, the number of SIP request roughly quadruples against base-line.

The following table summarized empirical impact of driving re-registration traffic to 180 seconds period for population of 1000 subscribers:

	Rate per second (incoming interface)
REGISTER requests (w and w/o digest)	20 pps
all requests	40 pps
all requests and answers	80 pps
bandwidth (TX and RX about 1:1)	380 kbps

This load is noticeable both in terms of bandwidth and CPU impact.

The following table present impact of media-relay on bandwidth for 1000 subscribers in peak periods. The underlying assumption is that in peak periods, there is one call for every ten active subscribers.

number of subscribers	1000
parallel calls (10:1)	100
G.711 bandwidth (RX and TX)	197*2*100= 39,400 kbps
SIP bandwidth (RX and TX)	380 kbps
Total bandwidth	~40 Mbps

7.8.4 Performance Tuning

The performance of the system can be increased by proper configuration of the hardware, operating system and the SBC. The following paragraphs list configuration suggestions that are known to bring the greatest performance benefits.

Hardware has expectedly profound impact on system performance. With networking applications, is network cards that shall receive particular attention. Our experience has been that Intel Ethernet cards are at least on part with cards of other vendors and often overperform them. Note that it is necessary that the kernel is using specific card drivers: performance of generic ethernet drivers is noticeably lower. Hints to hardware specific configuration option are provided in Sec. *Hardware Specific Configurations*.

Key card driver parameters that don't come preconfigured with the ABC SBC are those specific to Ethernet interfaces. If available, tune the following parameters:

- enable Receive Packet Steering
- increase coalesce and ring buffer size
- bond statically NIC RX queues to CPU cores

Furthermore, you shall also make sure that your ABC SBC configuration is not causing unnecessary load. Configuration options that can considerably increase overhead are especially media relay and registration processing. Media relay shall be avoided if not needed. If an SBC connects networks that are mutually routable, anchoring media may be entirely unnecessary. Also in many cases, when signalling passes the SBC twice on the way in and out, you may need to pay attention to configure the media to pass the SBC only once. Registration processing is primarily driven by the NAT keep-alive interval. We recommend a period of 180 seconds. Shorter intervals will not dramatically improve NAT traversal and will cost performance degradation. Longer intervals could result in expired NAT bindings for NATs that expire too rapidly.

Chapter 8

Monitoring and Troubleshooting

8.1 Overview of Monitoring and Troubleshooting Techniques

The ABC SBC and its accompanying monitoring product, ABC Monitor, are designed to provide real-time insight into service health and user behaviour for sake of troubleshooting, trending and security. Routine monitoring and troubleshooting is a key part of a SIP service life-cycle. It is also a complex one: the amount of traffic an SBC must handle is enormous and finding abnormal patterns in such quantity is not entirely easy. This is especially true when the service is exposed to a larger user population and is running on the public Internet. Also varying degree of SIP compliance of attached devices often causes unexpected behaviour.

Any abnormal service patterns can have a variety of reasons including unusual traffic caused by a security attacks or broken devices, or administrative shortcomings such as a incorrect rule-base or an under dimensioned system. Even if an abnormal situation does not impact a SIP service as whole but only a particular user it is important to find out what is happening.

Identifying presence and root causes of abnormal situations therefore requires solid data about the operation of the service. Here a virtue of the ABC SBC comes in play: it produces a lot of data reporting on the status of operation. In fact the number of bytes produced for monitoring typically exceeds the number of bytes used for the actual SIP signaling. What may seem disproportional is *the* recipe for the capability to understand and keep the status of operation smooth at any time. Good operational decisions can only be made with reliable intelligence.

In the following chapters we will discuss various methods how to monitor an ABC SBC-powered SIP service operation.

The most detailed and therefore powerful method to monitor the operation is using the **events** produced by the ABC SBC (if the event license is installed). The ABC SBC "documents" what SIP users are doing by issuing a report called event on every important user activity: registering, unregistering, failing to authenticate, completing a call, and so on and so forth. An administrator can even produce his own custom events. The events provide a history of user activity which can be looked backed at and analyzed. In a way, it tries to act as secret police would: it holds "files" on the observed subject that include an exhaustive gap-free activity history. At the same time, the overall collection of events also provides aggregated insights into the overall service health and can be used for example to see how the service usage varies in course of a day. The events are described in the Section *Events (optional)*.

The events do indeed come in a quantity that may make nailing down a problem or identifying a trend a tedious task. Therefore the ABC Monitor is available from FRAFOS to aggregate and filter the events. Using the ABC Monitor is documented in the section *ABC Monitor (Optional)*. In addition to user events, the ABC Monitor also shows the utilization of the system. If a situation requires, the ABC Monitor collects even traffic bits: SIP or even RTP data passing the ABC SBC. This is explained in the section *Diagnostics Dashboard*.

The next chapter, *Using SNMP for Measurements and Monitoring* shows how to monitor the overall system health using SNMP. SNMP is the industry standard for monitoring system health and is supported by many third-party monitoring tools, both commercial and open-source. The FRAFOS ABC SBC reports various OS-related and SIP-related counters using SNMP and can also report custom-based ones.

Additional diagnostic information is available directly in the SBC GUI. There is real-time GUI view of established calls and cached registration entries described in Section *Live ABC SBC Information*. There is also a possibility to review most recent traffic at IP layer as described in Section *User Recent Traffic*.

Additional methods for determining service status data are eventually described in the Sections *Command-line SBC Process Management* and *Additional Sources of Diagnostics Information*.

8.2 ABC Monitor (Optional)

The ABC Monitor provides administrators with an aggregated view of user activity based on usage reporting data collected from the ABC SBC/WebRTC gateways. This highly interactive, near real-time view can be used for trending, analysis of both short-term and long-term use patterns, troubleshooting, auditing server policies and identifying misconducting users. The reporting data comes using events from inside of the SBCs. This "insider view" allows the ABC Monitor administrators to inspect SIP traffic encrypted on the way from and to the SBCs, correlate calls "separated" by topology hiding, and report internal ABC SBC context such as traffic shaping decisions.

If there are multiple SBCs organized in a hot-standby pair, or a cloud the ABC Monitor will collect data from all of them and its centralized nature provides a global view of the whole system. An ABC SBC may also send its data to two Monitors in parallel. This is useful for various organization with multiple isolated teams, easy-to-start virtualised trials and migration scenarios.

The ABC Monitor user interface is organized in several dashboards. The opening Home Dashboard shows the most important data in a single comprehensible page, such as shown in Figure *ABC Monitor Home Dashboard*. All of the data relates to the period of time chosen in the top right corner. This page can also be sent to administrator on a daily basis by email to report on previous 24 hours.

The data in the home dashboard is structured in several rows. The first row shows various call metrics, such as number of completed and attempted calls, total number of minutes, etc. The second shows how frequent were events of the various types in the observed period of time. Dark fields represent many events of a kind. In this example we see that greylisting events were dominating at a time slot, a situation that often occurs when a SIP scan is launched on a public SIP service.

The next two rows show history of number of parallel calls and registrations, also compared with data from previous day shown using a thin line. In the boxes on the right hand side there are current numbers.

The last row shows number of security-related events. The timeline is divided in buckets and the number of events relates to each of the buckets. The bucket length grows proportionally with the time window. The number on the right-hand side shows number of security events in the most recent bucket. The number's background color changes with the number and is green when smaller or equal to five, orange bellow ten, and red above.

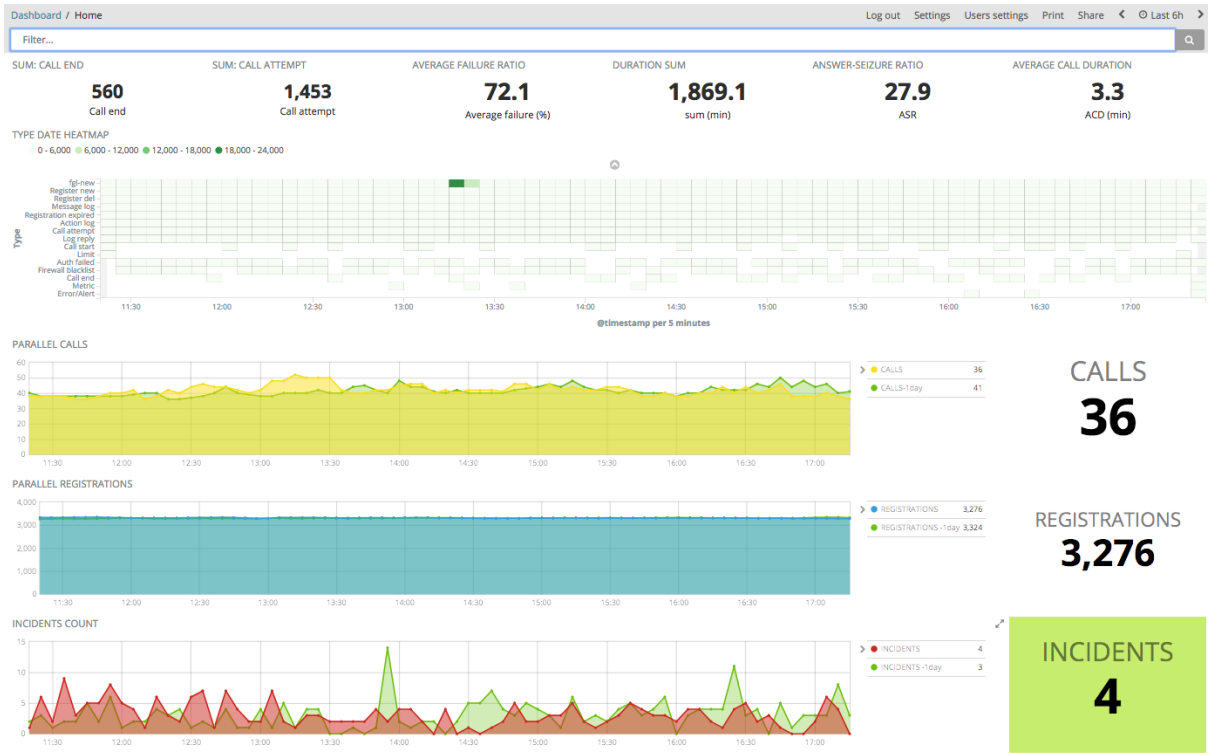


Fig. 1: ABC Monitor Home Dashboard

A snapshot of the home dashboard can be produced and sent as PDF attachment by Email every morning if a recipient email address is configured under "Settings → Reports: e-mail address for everyday reports".

All other dashboards are similarly organized. While they are organized along different aspects of SIP operation and show therefore different data, the visual structure follows the same pattern. In the very top, there are filters that allow to limit the events based on various criteria. Using the filters is described later in Chapter *Using Filters*. In the mid part, there are various graphical elements showing either some aggregated values, or their history over time. In the very bottom, there is a list of the actual events. The events can be clicked on to unfold and view all details.

The most frequently used dashboards are the following:

- "Call Dashboard" shows history of calls, analysis of failures and QoS reports. This is helpful to identify call trends, volumes, reasons of call failures, and troubleshooting specific calls. It provides both aggregate view of the situation as well as the possibility to review call details. More details are shown in Chapter *Calls Dashboard*.
- "Security" shows all Security events, also organized by the most active IP addresses and geo-locations.
- "Toplists" shows most active users by various metrics: call attempts, call minutes, number of short calls, etc.
- "Exceeded Limits" helps to alert on abnormal situations, such as excessively many phone calls from a single IP address. See Chapter *Exceeded Limits Dashboards* for more details.
- "Overview" holds all events relating to a user. There are really many. This is mostly useful when an administrator begins to suspect a problem and wants to see full user's history. Often administrators get alerted on a user when reviewing security dashboard, toplists, or exceeded limits, set up filter for such user, and inspect then his full event history. Additional details are shown in Chapter *Overview Dashboard*.
- "Connectivity CA" operates at topology layer and shows how good peering between calls from one Call Agent to another Call Agent is. This helps to identify Call Agents with poor performance, and/or Call Agents that have troubles making calls with each other. See Chapter *Connectivity CA Dashboard*.

- "Registration" shows registrations: new, expired and deleted, which transport is being used, from which part of the world are the registrations coming, and what SIP equipment is being used. This dashboard is useful for troubleshooting SIP user's connectivity. More information can be found in Chapter *Registration Dashboard*.

There are even more dashboards that provide less aggregated data that is useful when trying to understand a low-level problem.

- "Diagnostics" collects troubleshooting information. If an SBC administrator chose to record PCAP files, WAV files, produce custom event, or an unusual OS situation is reported, it appears here. See Section *Diagnostics Dashboard* for more information. Diagnostic events relating to layer-3 and layer-4 are separated in "Transport" dashboard and provide useful information to detect frequent retransmissions, or various TLS handshake failures.
- "Connectivity" is a dashboard operating at URI level that shows the most active caller-callee pairs.
- "Network Statistics" shows low-level data such as number of parallel calls, active registrations, bytes sent and received, etc. See Section *Network and Statistics Dashboard*. "Realms Stats" is a subset of network statics, broken down by realms.
- "Systems" shows how SBCs are doing in terms of memory and CPU. Useful to identify overload situations. See Section *System Dashboard*.

8.2.1 Events (optional)

Note that producing events is an optional feature that requires an additional licensing option and the ABC Monitor software.

The events collect a detailed history of user activities. With this data, it is possible to review in detail the history of a specific user as well as the whole SIP service. The events are produced by the ABC SBC in course of processing SIP and RTP traffic whenever some relevant action occurs: a call was attempted / established / terminated, a bandwidth threshold was applied, etc. Administrators may also choose to generate their own custom events.

The events are keyed by SIP address and IP address so that history of specific users can be easily established.

All events include three types of fields: mandatory, type-specific and call variables.

Every event includes mandatory fields identifying which SBC reported on which activity and when ("timestamp", "event-type" and "sbc"). The timestamp signifies the instant of time when an event was received by the ABC Monitor. This helps to overcome situations with multiple SBCs with unsynchronized clocks or different time-zone and may be slightly different than SBC-recorded time. A typical time sequence of events for a call is shown in Figure *Event of Call Timelines*: when an INVITE is arrived and PCAP file is recorded, "message-log" appears instantly. "Call-attempt" event appears as soon as a positive final answer comes back. Eventually "call-stop" appears upon receipt of a BYE from either party.

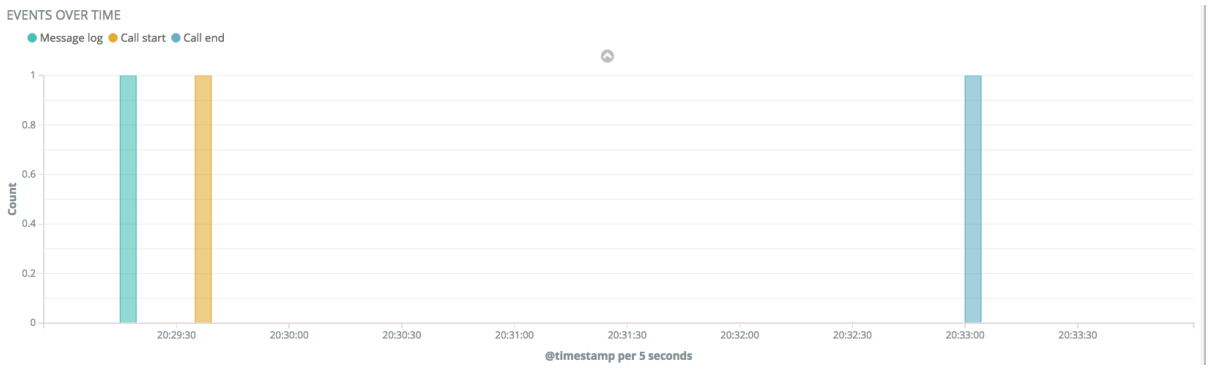


Fig. 2: Event of Call Timelines

Events relating to a SIP message include SIP addresses ("attrs.from", "attrs.to") and source IP address ("attrs.source"). Call-end events include "attrs.duration" expressing call length in seconds, whereas reg-new events

show the address of a newly registered SIP device in "attrs.contact".

Last but not least, script processing variables can be passed along with the call processing events – this can be for example useful to "label" the events with "tags" assigned to a call during call processing, such as "domestic", "long-distance" or "emergency". Some visualizations in ABC Monitor specifically require that an administrator sets well-known variables, as shown in Figure *ABC Rule for Setting a Destination Country Code by Request URI Prefix* and *Setting Minute Counter Call Variable*.

The ABC Monitor also enhances the events by additional fields used for security level assessment, geographical location, QoS information, and other data used for further analysis.

The following table shows content of a call-start event that is always produced when a successful INVITE transaction sets up a call. Internal fields with a leading underscore in name not shown.

Field	Value	Comment
@timestamp	2016-03-31T12:17:02.000Z	GMT event timestamp
@version	1	internal version number
type	call-start	event type
attrs.call-id	65601f8e625e0fb6484 ...	SIP callid of the call
attrs.dst_ca_name	pstn_gateway	name of Call Agent to whom the call is forwarded
attrs.dst_rlm_name	sipgate	name of destination realm
attrs.sbc	3e440ca4-00ee	id of the reporting SBC
attrs.src_ca_name	users	Call Agent from which the request came
attrs.src_rlm_name	public	realm from which the request came
attrs.from	sip:0000@172.27.10.114	SIP From URI
attrs.from-ua	VQM 0.4	User Agent Client type
attrs.method	INVITE	SIP Request method – always INVITE for call-start
attrs.r-uri	sip:echo@free.tel	SIP Request URI
attrs.sip-code	200	Numerical code of SIP reply always 200 for call-start
attrs.sip-reason	OK	Human-reasonable reason phrase in SIP reply
attrs.receiver_ip	192.168.0.111	SBC's IP address at which it received the INVITE
attrs.ruriip	free.tel	host part of request URI
attrs.scenario	call	this is a scripting variable chosen to be passed along with the event
attrs.source	10.0.0.10	source IP address of the request
attrs.src-port	1085	source port number of the request
attrs.to	sip:echo@free.tel	To URI as in the INVITE request
attrs.to-ua	F-PBX 2.3	Signature of the called party's UA Server
attrs.transport	udp	transport protocol used for signaling
id	483B139F-56FD153E...	internal ID. useful for correlating multiple related events

The rest of this Section is structured by the event types that the ABC SBC produces:

- *Call Processing Events* – these are events that describe SIP calls and are mostly used to observe user habits, reasons for call failures, and QoS
- *Registration Events* – these are events that describe how SIP devices register with the SIP service. The events show the reachability of the SIP users in time.
- *Diagnostics Events* – these events help to identify unusual traffic patterns, misconfiguration of the service and other irregular situations.
- *Security Events* – these events report on SIP traffic which may possibly indicate attempts to compromise security of some SIP users or the SIP service as whole

Call Processing Events

These events are generated automatically at different stages of the SIP call establishment process, see Fig. *SIP call processing events*.

- call-start: generated after a successful call establishment. The method is always INVITE and sip-code 200.
- call-attempt: generated after an unsuccessful attempt to establish a call due to caller canceling the call, callee declining it, or a timeout. Failed authentication attempts are reported on in separate events. The events always include the SIP code with which the call attempt was rejected.
- call-end: generated after an established call is terminated. They include a full report on how the call completed. The From and To event fields take the same values as call-start event – they signify who initiated the call (and not who initiated the call termination). The event-specific fields include:
  - The field "originator" specifies who caused the call termination and can take the following values: "caller-terminated", "callee-terminated", "call-length-terminated" (SBC terminated upon exceeding the maximum call length limit), "no-ack" (SBC terminated due to missing ACK), "rtp-timer-terminated" (SBC terminated upon RTP inactivity), "session-timer-terminated" (SBC terminated upon session timer expiration), "admin-control-terminated" (administratively terminated from GUI), "internal-disconnect" (call terminated due to a internally transferred call), "reply" (negative response received on an established dialog: 404, 408, 410, 416, 480, 482, 483, 484, 485, 502, 604), "server-shutdown" (server process terminated due to a SIGUSR1 or SIGUSR2 signal), "srtp-failure" (SRTP key negotiation failure), "internal-error" (internal error).
  - The field "duration" specifies the length of call in seconds.
  - The fields "rtp-stats-a" and "rtp-stats-b" represent the RTP statistics for the media streams on each call leg. Each call leg contain one or more media streams, each of which offer incoming and outgoing information.

The following table shows the information for the incoming media streams.

Field	Value	Comment
ssrc	413934793	incoming SSRC value
src_ip	192.168.0.155	source IP address
src_port	37454	source port
dst_ip	192.168.0.155	destination IP addres
dst_port	46920	destination port
payload	PCMU/8000	media payload
packets	52832	received packets
bytes	9087104	received bytes
last_seq_nr	54428	last received sequence number
max_delta	3	maximum delta between two packets
max_delta_seq	21397	sequence number of the packet with the maximum delta
max_burst	69	maximum number of packets per second
lost_percentage	0	lost percentage
jitter	6	jitter
dropped	0	packets dropped
seconds_since_last_received_packet	0	seconds since last received packet

The following table shows the information for the outgoing media streams.

Field	Value	Comment
ssrc	473964392	outgoing SSRC value
src_ip	192.168.0.149	source IP address
src_port	27054	source port
dst_ip	192.168.0.155	destination IP addres
dst_port	26120	destination port
payload	PCMU/8000	media payload
packets	52832	received packets
bytes	8710432	received bytes
last_seq_nr	24326	last received sequence number
lost_percentage	0	lost percentage
rtt_min	5	minimum round trip time
rtt_max	172	maximum round trip time
rtt_avg	26	average round trip time
jitter	6	jitter
seconds_since_last_sent_packet	0	seconds since last sent packet

The call-end and call-start events have the same ID which can be used for correlation. This is however more often used for correlation with other events, like recording for example, because there is no additional data in call-start beyond call-end.

From-URI	To-URI	SIP callid	Result code	Response phrase	Duration	User agent (from)	User agent (to)	AoR	Contact
sip:alice@test.com		w0PtInpA3u3xqXGLOI0C60COqPWR3Ifb		ctrl-cmd	00:00:20				
sip:alice@test.com		3zoHtsOwykozxiw2MZU5cXpBDNTTzFrl		timeout	00:01:29				
sip:bob@test.com	sip:alice@test.com	5002B22A-51B99C4900013F38-F327A700		bye	00:00:05				

Fig. 3: SIP call processing events

An important capability is passing call variables (Section *Binding Rules together with Call Variables*) in the call processing events. This can be used to "tag" calls with various hints, for example to discriminate domestic and international calls. In some cases, there are predefined call variables that have a specific meaning to the ABC Monitor. Such are currently **dst_cc** (Figure *ABC Rule for Setting a Destination Country Code by Request URI Prefix*) and **minute_counter** (Figure *Setting Minute Counter Call Variable*).

To turn variable passing on for call events, turn on the SEMS Global Configuration Option **Add call variables into events**. Care needs to be applied so that some possibly sensitive information is not passed along with the variables to the ABC Monitor.

The call variables appear in the events with the "attrs." prefix. If a variable name is the same a name of a mandatory event field, the call variable overrides the field value. This can be useful for example when identity of a SIP user should be presented in a different way (altered, anonymized) than seen in the SIP traffic. We however recommend to use this overriding capability with caution as it leads to the loss of the original traffic view. Example of such a rule and the resulting event is shown in Figures *A Rule for Rewriting an Event Field with a Call Variable* and *An Event with a Field Modified by a Call Variable*.

Fig. 4: A Rule for Rewriting an Event Field with a Call Variable

Fig. 5: An Event with a Field Modified by a Call Variable

Registration Events

The registration events are generated automatically at different stages of processing SIP REGISTER requests when register caching is enabled (see Section *Registration Caching and Handling*).

- reg-new. This event is produced when a SIP User Agent registers a new contact through the ABC SBC.
- reg-del. This event is generated when a SIP User Agent deregisters a previously registered contact using the RFC3261 procedures. This is typically the case when a softphone is shut-down and it unregisters gracefully. Some clients are also implemented a way that they unregister and re-register newly instead of periodically renewing one registered binding. See an example of such in the ABC Monitor snapshot in Figure *Event Timeline for a SIP Device Registering and Unregistering Periodically* – unregistered bindings are immediately followed by new registrations.

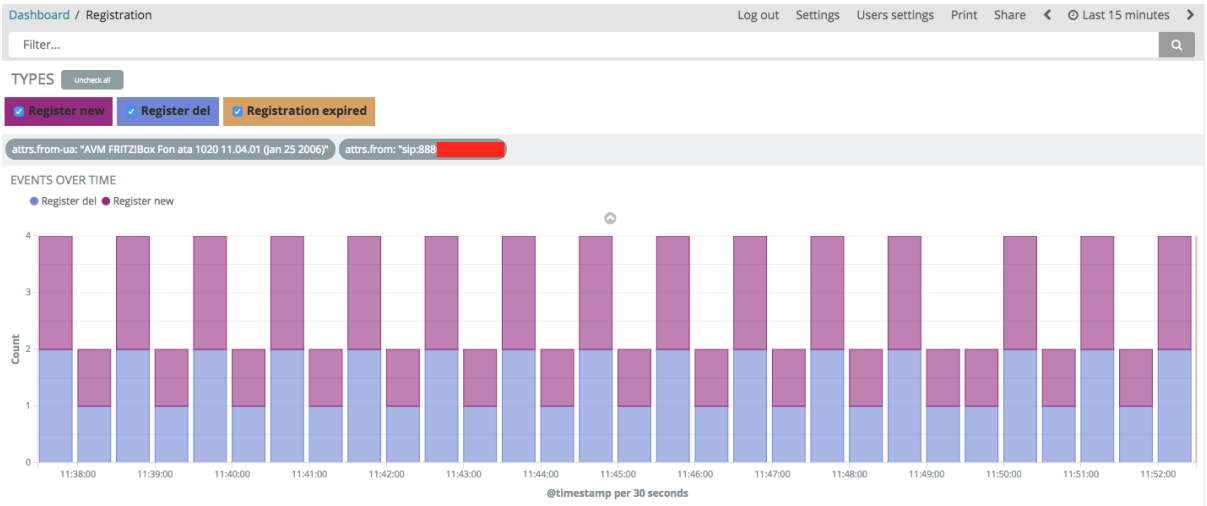


Fig. 6: Event Timeline for a SIP Device Registering and Unregistering Periodically

- reg-expired: Indicates an expired registration binding. This happens if an upstream SIP client fails to renew its contact within the window re-registration window agreed upon between the ABC SBC and downstream registrar. When this timer expires, the binding will be deleted and no incoming requests can be forwarded using the particular binding. This often happens with clients that don't comply to RFC 3261 by not respecting the server-side-imposed registration renewal

interval or vary the contacts inadequately. Example of a timeline for such a devices is shown in Figure *Event Timeline for a SIP Device Failing to Re-register Timely*. Such a devices remains unreachable in the periods of time between the orange expiration and green re-registration bars.

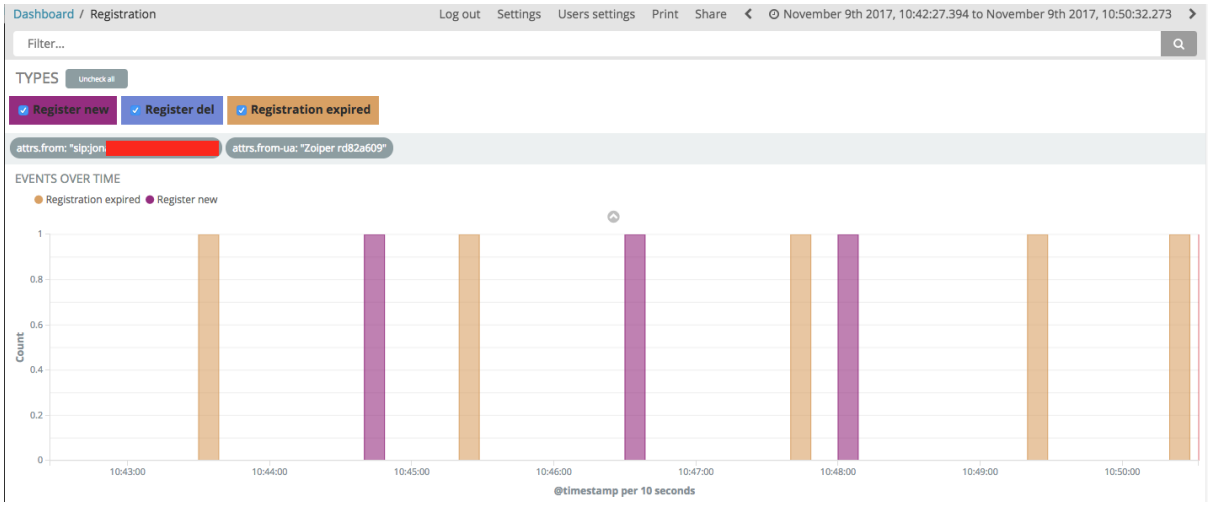


Fig. 7: Event Timeline for a SIP Device Failing to Re-register Timely

Diagnostics Events

The diagnostic events are used to identify conditions that provide additional diagnostics information and sometimes alert on conditions an administrator shall verify whether they are normal. These events are triggered by minor errors, completed playing of or recording of WAV/PCAP files, change in status of transport blacklisting, and custom events.

- *Custom events* (action-log): This is one of the most important diagnostic events available in the system. It is triggered from the ABC SBC rule-base using the "Log Event" rule action. Typically it is used when an administrator wants to see if a specific rule is indeed evoked and how often. Also administrators use the custom events when they begin to suspect some undesirable traffic and don't want to drop it yet. This action allows them to observe the suspicious traffic before making a further action. The conditions can be any that the rule definitions allow and often includes tests if a SIP device is registered, shows a suspicious User Agent type, tries to call a premium phone number or otherwise falls in the "suspicious category". For example an administrator may choose to observe all SIP messages that come to the ABC SBC without a username in To header field URI. Example of such a "Log event" rule is shown bellow in Figure *Rule for Reporting on SIP Requests with Empty Username in From*.

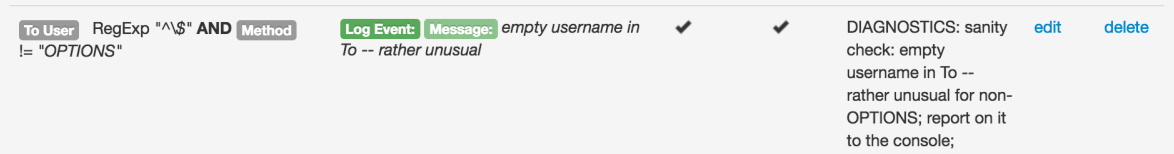


Fig. 8: Rule for Reporting on SIP Requests with Empty Username in From

The action includes a parameter where the administrator can specify additional text describing the event. The parameter can include replacement expressions providiong additional information about the processed SIP message. These should be used only if necessary – when varying elements are present in event description ABC Monitor software cannot group the events by the same description.

- *recording events* (recording): These events are generated if voice recording was enabled for a particular call. The events include HTTP reference to a file with the recorded WAV file. See also Section *Audio Recording*.

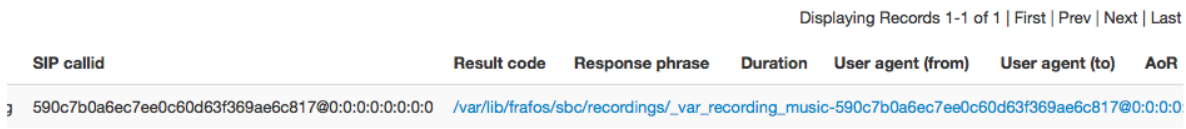


Fig. 9: Recording Event

- *SIP traffic logging events* (message-log): These events are generated only if a rule was set up to record SIP/RTP traffic using the **Log received traffic** action (see Section *Diagnostics Dashboard*). The event includes references to the recorded PCAP file and a ladder chart displaying the recorded traffic. The PCAP files appears always later than the event: the event appears when recording begins, the file when it recording ends and upload to ABC Monitor completes in background. Figure *Ladder Diagram for the Suspicious User* shows an example of such ladder-chart rendered by the ABC Monitor and showing both sides of a SIP message – incoming and outgoing.
- *Error/Alert* (error): These events are always produced when no route matches for a SIP request, a TLS connection is refused, terminated or other unspecified error occurs. The ABC SBC reports these because TLS credential management is often misconfigured and needs to be fixed for TLS clients to be able to connect. Alerts may also appear if the system is under-dimensioned and include messages like "/data disk usage above 80%" or a misconfiguration has been encountered in rule-base that was detected run-time (for example: routing failed: can't parse outbound proxy URI: 192.168.0.85).
- *Notice* (notice): These events are produced when some layer-3 or layer-4 conditions change. This is currently the case when a TLS connection opens or closes successfully, or when health status of a Call Agent changes so that it is either removed from or added to a transport blacklist. (See Section *IP Blacklisting: Adaptive Availability Management*). *Too many SIP retransmissions event* notice events are generated if a SIP transaction reaches the defined number of retransmissions. The retransmission number triggering the event is globally configured under **"Config → Global Config → Events → Generate an event if a SIP transaction reaches .."**. The default value is 0, which disables the event notification. Notice events are also produced when traffic reaches a shaping soft-limit (see Section *Traffic Limiting and Shaping*).
- *Prompt*: These events are always produced when a caller's attempt is handled using local audio announcements (see Section *Playing Audio Announcements*).
- *dest_monit*: The ABC SBC reports these events when availability monitoring is enabled for a Call Agent (see Section *IP Blacklisting: Adaptive Availability Management*). In ABC Monitor the events are visualized in the CA Availability diagram in the "Connectivity CA" dashboard as shown in Figure *Call Agent Availability Lanes*.

Security Events

This Section discusses events that have relevance to security of a SIP service. These security events are generated when messages are dropped because of failing to accommodate a security policy. This can be because the traffic has exceeded traffic limits, a drop action has been applied, authentication failed or an unfavorable SIP answer came from downstream in response to a SIP request.

Counter-measures to fend off security attacks are discussed in a separate Section *Securing SIP Networks using ABC SBC and ABC Monitor (optional)*. The event types are the following:

- **limit**: These events are generated if some of the traffic constrains (see Section *Traffic Limiting and Shaping*) has been exceeded. For example an administrator may choose to ban signaling traffic from an IP address if it sends more than 10 requests per a minute. See Figure *Limit Events* for example of limits reporting on traffic shaping in effect. The limit event type is also generated when current traffic volume exceeds limits set by the ABC SBC software license.

SIP callid	Result code	Response phrase	Duration
c76553ac-14cbc0ad-6772cee2@192.168.178.29		RTP bandwidth limit towards callee reached	00:00:00
c76553ac-14cbc0ad-6772cee2@192.168.178.29		RTP bandwidth limit towards caller reached	00:00:00
e1130136-ff37103f-5506921c@192.168.178.29		RTP bandwidth limit towards callee reached	00:00:00
e1130136-ff37103f-5506921c@192.168.178.29		RTP bandwidth limit towards caller reached	00:00:00
a8cad0100f70c974a8aaa908b07cf670@0:0:0:0:0:0:0:0		CAPS limit reached for counter 6ee06b93-06cc-84e8-fbc5-0000288d644f	00:00:00

Displaying Records 1-5 of 5 | First | Prev | Next |

Fig. 10: Limit Events

- **message-dropped**: These events are generated if a message was silently discarded using the *drop* action in A or C rules. Sender of the discarded traffic will not see any answer to his request. Note that if he is probing the service using TCP he will still be able to find out if there is running service. (Section *Manual SIP Traffic Blocking* provides more details on blocking traffic using the *drop* action).
- **auth-failed**: This event is triggered always when a SIP request authentication fails. Note that the way the SIP protocol works, an initial request is always challenged by server using the 401/407 replies. This initial challenge does NOT trigger the event. Only when the subsequently re-submitted request with credentials fails to authenticate and yields a 401/407 answer, the auth-failed event is generated. The reason why a request fails to authenticate may be multi-fold and needs deeper examinations. A SIP phone user may fail to configure his device with proper SIP URI and/or password. It may be administrative mistake on the server side such as deactivating a user account. However it may be also a password-guessing attack, such as shown in Figure *ABC Monitor Displaying How a Brute-Force Password Guessing Attack Ramps Up*. The sudden increase in number of authentication failure clearly indicates an attempt to breach security of the SIP service.

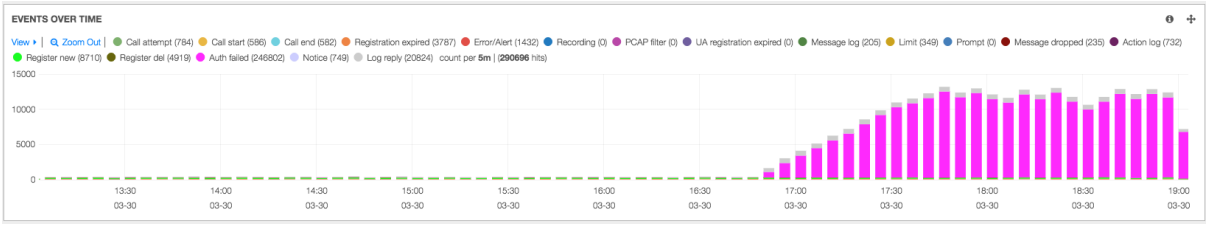


Fig. 11: ABC Monitor Displaying How a Brute-Force Password Guessing Attack Ramps Up

- **log-reply**: The log-reply event allows an administrator of the ABC SBC to identify traffic that apparently irritates downstream SIP equipment. If for example a downstream server chooses to send a 604 for requests that use non-existent SIP URIs, the ABC SBC may be configured to report on such using the *log replies* action as shown in Figure *Rule for Reporting All 604-replied Responses*. Similarly the events can be generated on receipt of any other specific reply codes, such as 403 (Forbidden).

Fig. 12: Rule for Reporting All 604-replied Responses

An example of events captured during a scanning attack is shown in Figure *Events Produced During a Scanning Attack*. In the "To" column you actually see that the attacker was trying to register under different

numbers beginning with 12: 122667, 12554, 122562, etc about every two seconds. When he hit a non-existing account (12554 for example), a 604 came back and triggered a "log-reply" event. However as he tried 12667, a 401 came back revealing to him that he had "pinged" an existing account, just without proper credentials.

Timestamp	Type	Reason	To
2016-03-31T18:36:30.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:36:29.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:36:28.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:36:20.000+02:00	log-reply	this From URI not served here	sip:12554@sip.
2016-03-31T18:35:51.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122562@si
2016-03-31T18:35:50.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122562@si
2016-03-31T18:35:50.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122562@si
2016-03-31T18:35:48.000+02:00	log-reply	this From URI not served here	sip:12563@sip.
2016-03-31T18:35:48.000+02:00	log-reply	this From URI not served here	sip:12561@sip.
2016-03-31T18:35:39.000+02:00	log-reply	this From URI not served here	sip:12556@sip.
2016-03-31T18:35:27.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:35:27.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:35:27.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:35:19.000+02:00	log-reply	this From URI not served here	sip:12554@sip.

Fig. 13: Events Produced During a Scanning Attack

- **firewall-blacklist**: These events identify blocked IP addresses. They are are generated when an ABC SBC chooses to drop traffic from an offending IP address. See Section *Automatic IP Address Blocking* for configuring criteria for automated IP address blocking in an ABC SBC.
- **firewall-greylist**: These events are generated when an ABC SBC chooses to drop traffic from an IP address that sent the ABC SBC some initial traffic but has not managed to establish trust. See Section *Automatic Proactive Blocking: Greylisting* for configuring blacklisting in an ABC SBC.

8.2.2 HOWTO Find a Needle in the Haystack: Iterative Event Filtering

The ABC Monitor combines both aggregated view of event data as well as the actual data details. This is instrumental in finding problems quickly: the aggregated view helps to detect a trend or anomaly which would be hard to find in the vast amount of SIP data. Once a situation worth further investigation is detected, the administrator can apply different filters consecutively until the root cause is identified using event details. The details can go as low as bits of SIP messages.

In this chapter, we will show an example how to iteratively proceed from detecting a high-level problem to finding the low-level bits triggering it. The examples are taken from a real operation and therefore many of the elements in the screenshots are shaded.

For example, administrator may find in Call Dashboard that average call failure ratio over 50% is too high, see that most frequently occurring call failure reason is 480 ("User Offline" in SIP specification), and start nailing the problem down by applying event filters.

This is the situation shown in Figure *Example of a High Call Failure Rate Situation*. Average failure rate is at 68%, the most massive error code is 480.

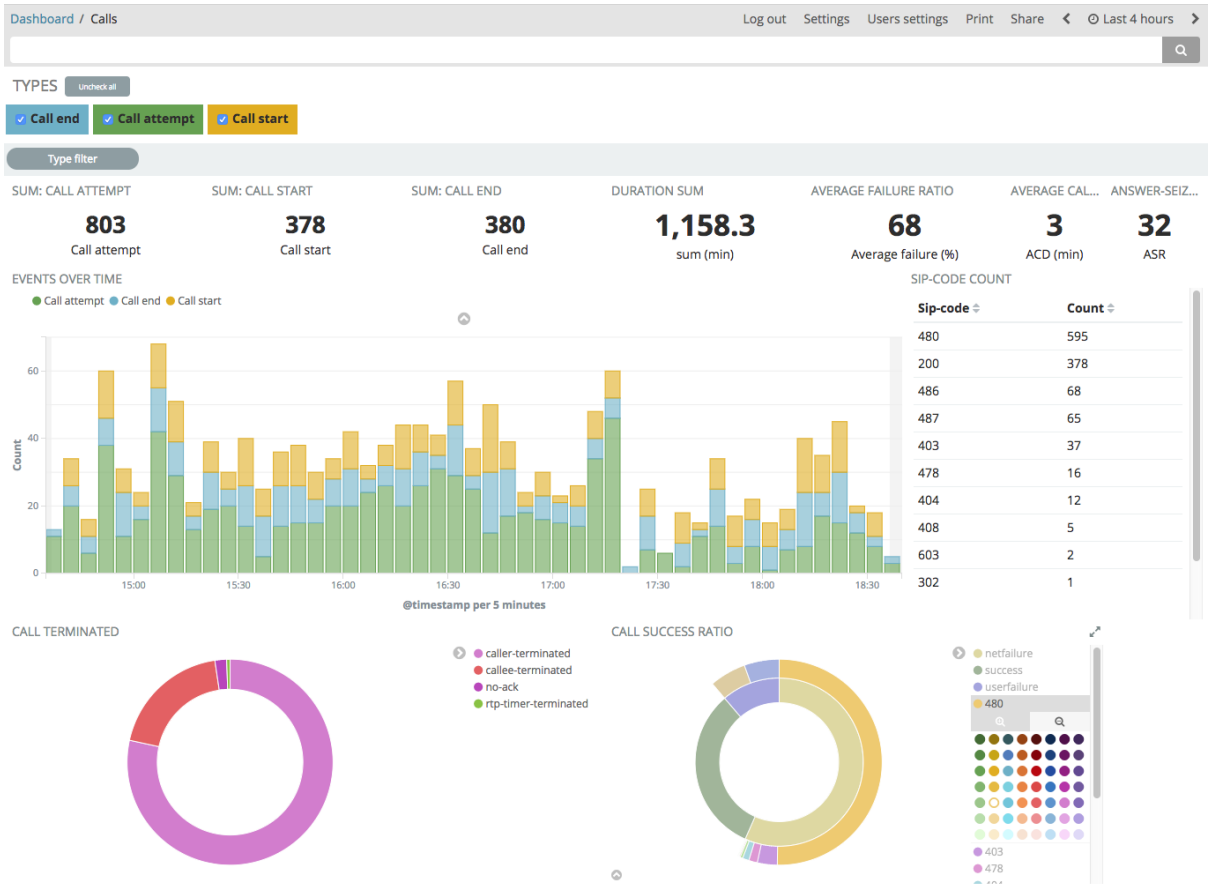


Fig. 14: Example of a High Call Failure Rate Situation

What the administrator typically does in such a situation is to spot the unusual trend, and inspect its details. Here the abnormality is the unusual number of 480s, in fact the typical top error codes are Busy (486) and Canceled (487). Therefore the administrator will limit the events to those with the SIP code of 480. He does so by clicking the "plus magnifying glass" icon underneath the 480 code, and "pinning" the filter using the pin icon in the top bar so that the filter can be transferred to a Dashboard like "TopLists". The filter looks like in Figure *Filter for Further Inspection of too Many 480s*. The statistics have changed because the filter limited inspected events only to call attempts failing with 480 code, and also because the time window has slightly advanced in the meantime.

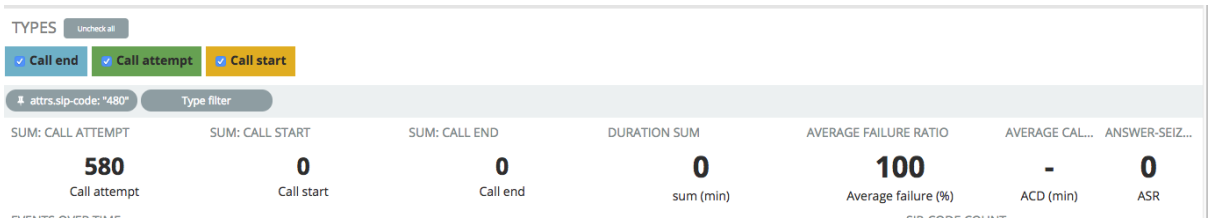


Fig. 15: Filter for Further Inspection of too Many 480s

When we now switch to the Toplist, we will find out that vast majority of the 480-terminated call attempts is coming from a single domain, and inside this domain an anonymous user is dominating. (Figure *The Top 480-er*) While we do not know, if it is a user trying to desperately reach an offline called party, or someone scanning calls, we can pin a filter by caller, deactivate the filter by error code and see the full history of the suspicious user in Overview.

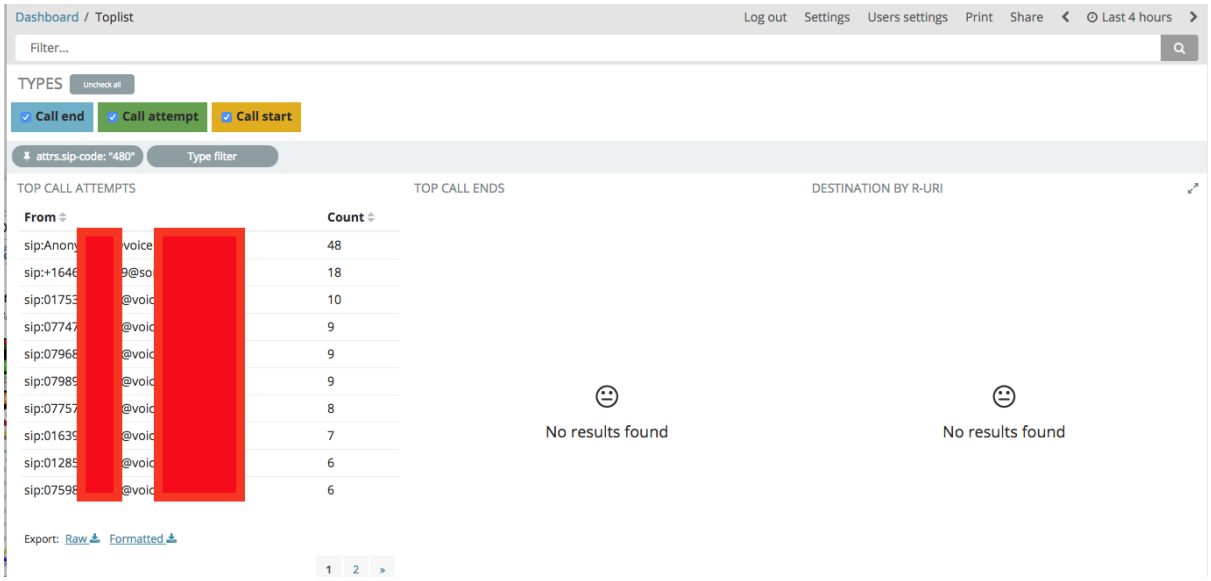


Fig. 16: The Top 480-er

The Overview gives us a picture of a suspicious user who keeps making call attempts at a high rate without success and also without attempt to register. This is seen in *Complete Event History of a Suspicious User*.

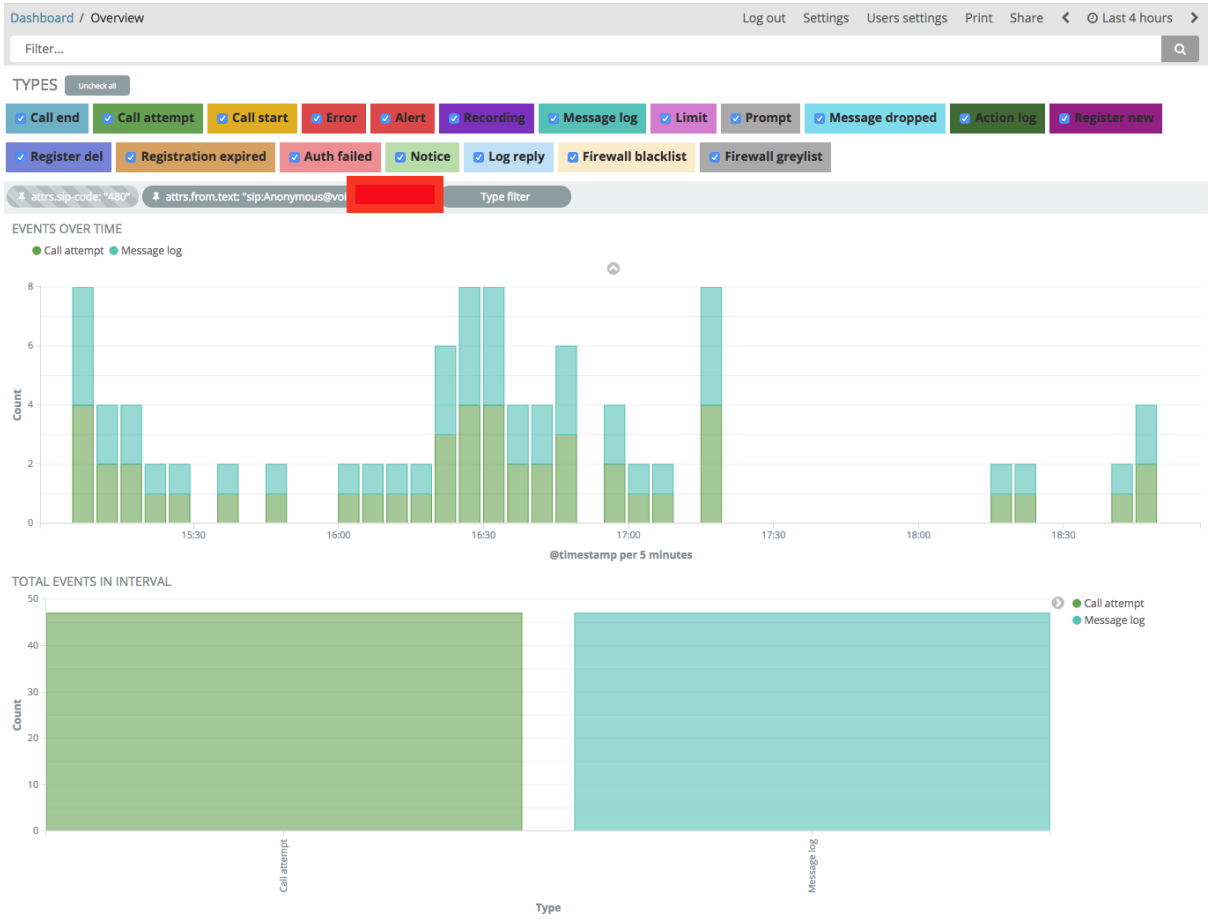


Fig. 17: Complete Event History of a Suspicious User

It pays off therefore to scroll down and look at event details for this particular user. Not only is there a detailed

report on the call attempt, but under “View Messages” there is a link to ladder diagram showing the actual SIP message exchange.

Fig. 18: Complete Event Details

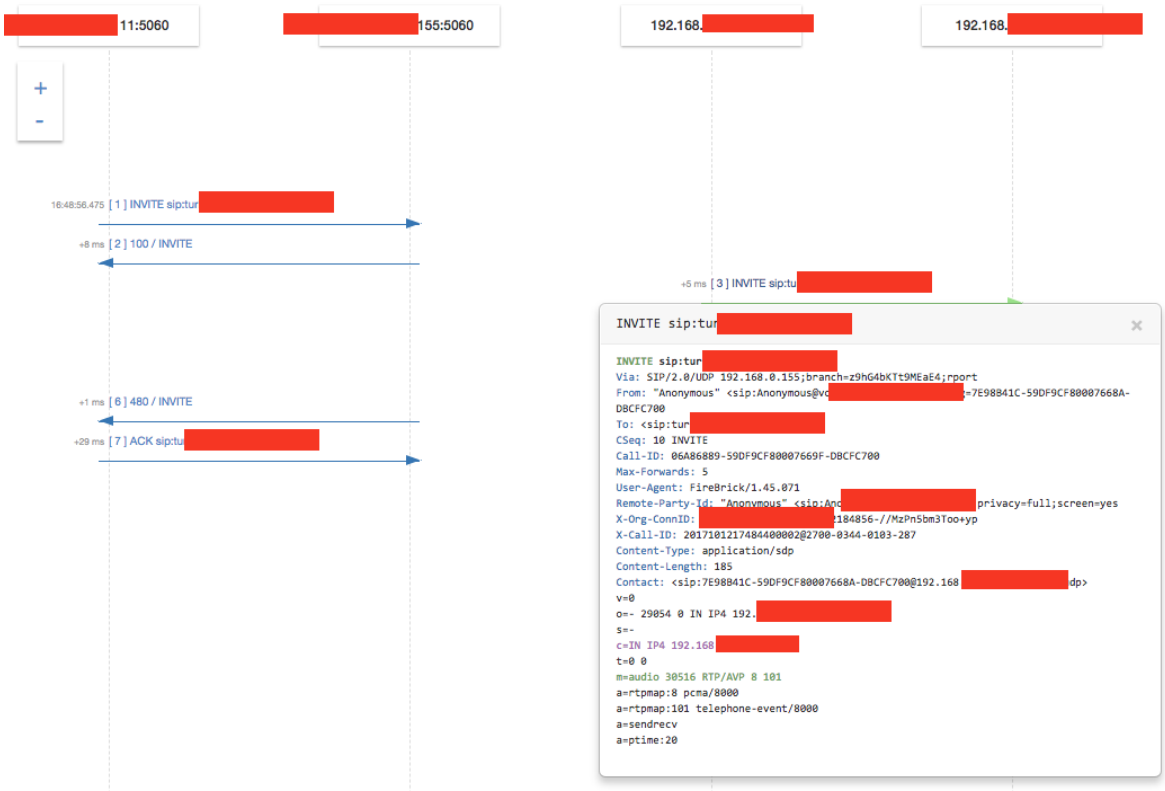


Fig. 19: Ladder Diagram for the Suspicious User

The ladder diagram shows the message flow and its timing, as well as details of SIP message, including From and To identities, type of SIP User Agent, and SDP media negotiation payload. The capability to view internal perspective of traffic is very valuable – if traffic is encrypted, it is difficult for a troubleshooter to inspect its content. Also if the SIP traffic is obfuscated by use of topology hiding (see *Topology Hiding*), it would be difficult to relate incoming to outgoing SIP dialogs without the internal perspective.

In summary, we have shown in this example how to detect unusual situations using aggregated views (too many 480s), filter out events specific to the situation, find a user that has caused most of them and inspect in detail his

gap-free history and even SIP message details. This iterative process gives every administrator powerful tools to find out what is going on in a SIP service, and have good information to decide if he is dealing with an abnormally active user, malicious attack or a network misconfiguration.

8.2.3 Using Filters

As shown in the previous chapter, filters are the essential instrument for finding out what is going on. In this chapter we describe all of the filter types available in ABC Monitor. There are data filters, type filters, time filter, and full-text filters. Multiple data filters can be combined, in which case events will be sorted out that match **ALL** of them. All of the filters appear in the most upper part of the dashboards. For example, events can be restricted to all but registration events, as long as they relate to an IP address, lead to a "480" SIP failure code and are for a given SIP user, as shown in Figure *Example of a Combined Filter*.

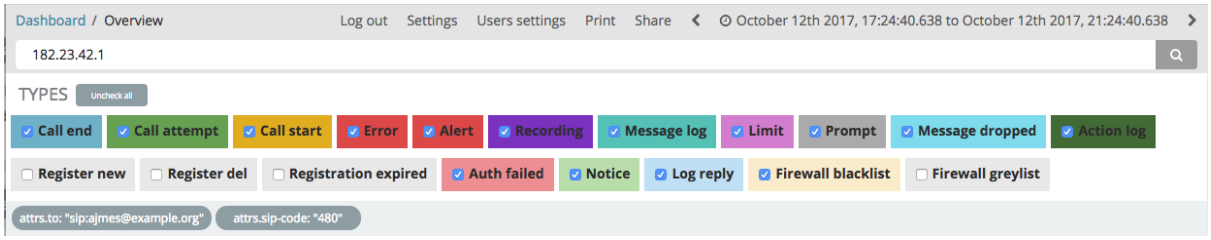


Fig. 20: Example of a Combined Filter

Data filters are used to filter out all events with the same data field values. They can be created from many elements shown in the dashboards. When user hovers over most of the data elements shown in the dashboards, two magnifying glass icons with plus and minus symbol will appear. By clicking on either icon, a filter is created that restricts events to those that either have (plus icon) or do NOT have (minus icon) the same value.

For example, one can visit the Call Dashboard, review the most frequent SIP error codes, and filter out call attempts relating only to 403 (Forbidden) as shown in Figure:

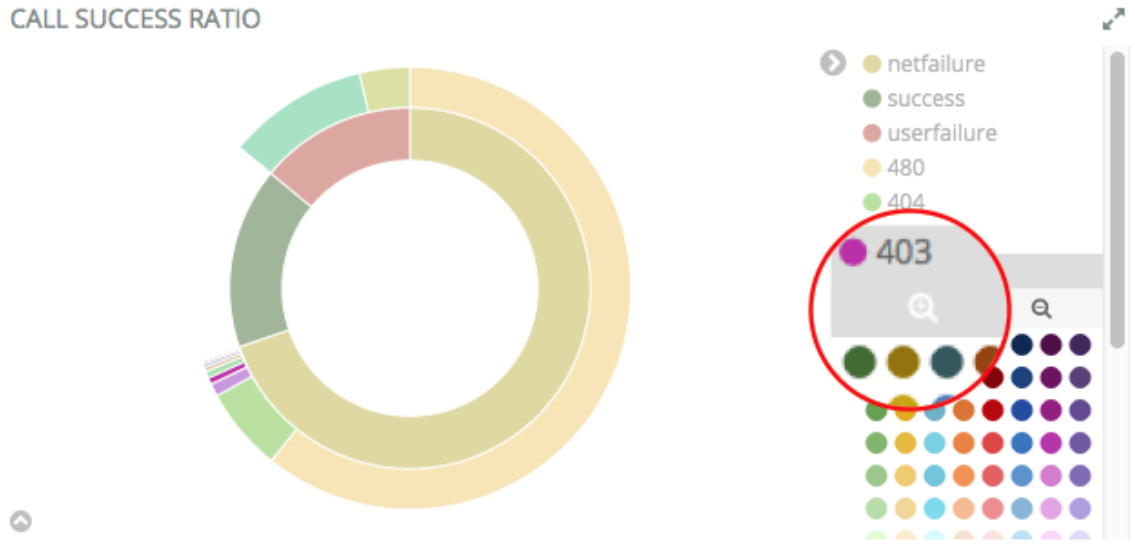


Fig. 21: ABC Monitor filtering out 403-ed call attempts in Call Dashboard

Every data filter can be deleted, deactivated, and importantly pinned – as shown in the example in the previous chapter, a pinned filter can be transferred to another dashboard where some specific aspect of the filtered data is easier to find. To pin a filter, hover over it and click on the pin icon. Unpinning is done the same way.

Other possibilities to adjust filters include temporary deactivation using the checkbox icon (filter appears then dimmed), permanent deletion using the thrash bin icon, and filter negation using magnifying glass icon (negated filters appear in red).

Fig. 22: Filter Alternations: pinned, pinned and negated, deactivated

Type filters checkboxes are shown in the top-bar and allow to easily restrict events by their respective types. This is particularly useful in dashboards with many event types, like in Overview, when administrator wishes to filter out events unrelated to his case.

Time filter sets the window of inspect time either absolutely, or relatively to current time. Of course, it can only cover the time period for which events are stored, as configured during the ABC Monitor installation.

Last but not least, the top search box allows to add **full-text filter** that looks for a pattern in multiple fields of the available events. By default, the full value, such as CallID must be included. Special terms can be used as follows:

- *, asterisk, stands for a wildcard and can substitute for any number of any characters. Use of wildcards slows down the search.
- \, backlash, means that the subsequent character will be interpreted literally
- a **colon**-separated <name>:<value> pair means that the searched value is looked for only in a field of the given name
- combinations of the terms are possible: **AND** allows to introduce multiple conditions, all of which must be met; **OR** allows to introduce multiple conditions, any of which must be met; **NOT** allows negation
- also the syntax "attrs.source:[from_ip TO to_ip] allows matching the event source IP address against an IP address **range**

Therefore if there is a user Wesley making calls using his SIP address sip:wesley@frafos.net to reach the SIP address sip:123456@example.net and he makes the calls from an IP address 192.168.0.85 belonging to the Call Agent "wesley-net", the following search expressions will match:

- **sip\:wesley@frafos.net** will match all calls from/to Wesley; note that colon must be preceded by backslash, otherwise the ABC Monitor would attempt to search through a field named sip
- ***wesley*** will match all previous records, and probably some more as well, such as wesley.home and wesley.office. It will also match all calls from and to the Call Agent "wesley-net".
- **attrs.dst_ca_name:wesley-net** will match all calls towards the Call Agent **wesley-net**
- **192.168.0.85** will match all events relating to that IP address.
- **attrs.source:[192.168.0.0 TO 192.168.0.255]** will also match because Wesley's IP address is inside the range
- **487** will match all call attempts that failed with 487 SIP code
- **attrs.sip-code:487 OR attrs.sip-code:486** will match all call attempts that failed because of 487 (cancelled) or 486 (busy)

The following search expressions will not match:

- **wesley** will not match, because full-match is attempted without wildcards
- **sip:wesley@frafos.net** will not match because of the colon
- **NOT attrs.source:[192.168.0.0 TO 192.168.0.255]** will certainly match many events but not Wesley's as his IP address is in the negated IP range
- **attrs.duration:[500 TO *]** will filter out all calls exceeding 500 seconds in duration

8.2.4 Overview Dashboard

The Overview Dashboard displays events of all types. This is often used, when inspecting a gap-free history of a specific IP address or user identified by a URI.

In the following example, we look at traffic generated in the frafos.net domain. We let a user to register using wrong password (failed-authentication event), then retry using a correct password (register-new), make a call to an announcement (prompt, and also message-log because administrator chose to store all SIP traffic on this SBC, and action-log because administrator chose to issue a custom event for calls to a specific destination).

Fig. 23: Example: gap-free history of all events in a domain

Some other interesting chart in the Overview dashboard is that depicting total number of events by time. Especially finding a disproportionally high number of a specific event type indicates an unusual situation. For example a high number of greylisting events failures as shown in Figure *Total Events with Disproportionally High Number of Greylisted IPs* typically signifies a security attack.

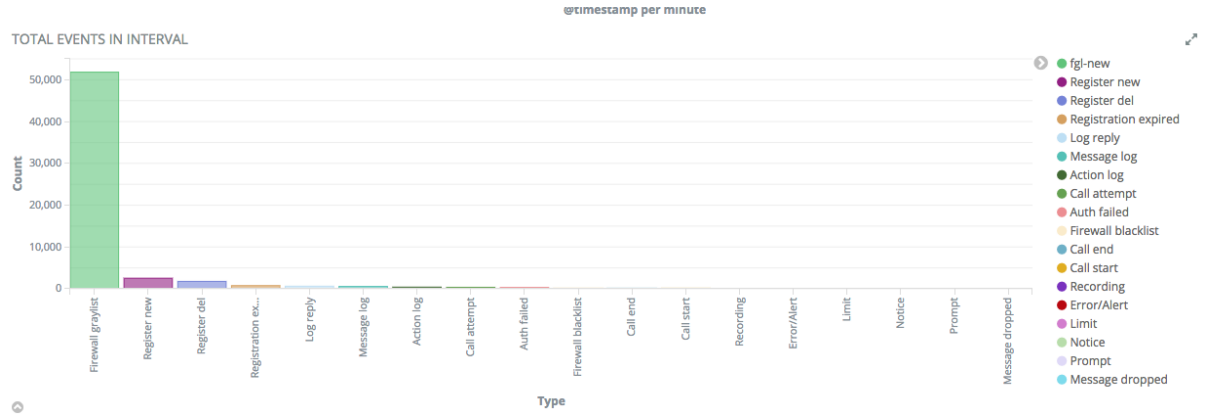


Fig. 24: Total Events with Disproportionally High Number of Greylisted IPs

8.2.5 Calls Dashboard

The Calls Dashboard analyzes call-related events (see Section *Call Processing Events*) to summarize processed SIP calls and how their quality was.

A screenshot of the top part of the dashboard has been already shown in Figure *Example of a High Call Failure Rate Situation*. From top to the bottom, there are call statistics, call events timeline, and breakdowns of successful calls by termination party and final status.

The pie chart breakdowns help to identify in detail why calls are being terminated. The left-hand side pie chart shows who terminated established calls. The normal termination types are "caller-terminated" and "callee-terminated". However calls could have been also terminated by the ABC SBC for a variety of reasons. These include "no-ack" when a caller failed to deliver the SIP ACK request, "rtp-timer-terminated" when RTP media stopped flowing without clean SIP session termination. See the Section *Call Processing Events* for the full list. The right-hand side pie chart shows both successfully established calls and failed call attempts and structures them by status code in the outer ring. The status codes are categorized in the inner circle intro three groups: success (200-answered INVITEs), userfailure (486 Busy and 487 Canceled) and network failure (everything else). Clicking on a pie chart segment allows to introduce a filter for the events of the same kind.

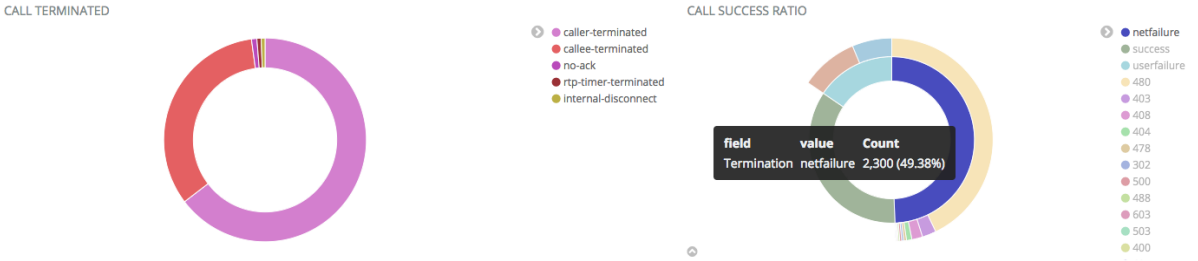


Fig. 25: Call Completion Status Breakdown

The lower dashboard part is shown in Figure *Screenshot: Lower Part of the Call Dashboard* and includes call durations, break-down of calls by countries, and eventually quality details of QoS-troubles calls and the call event details.

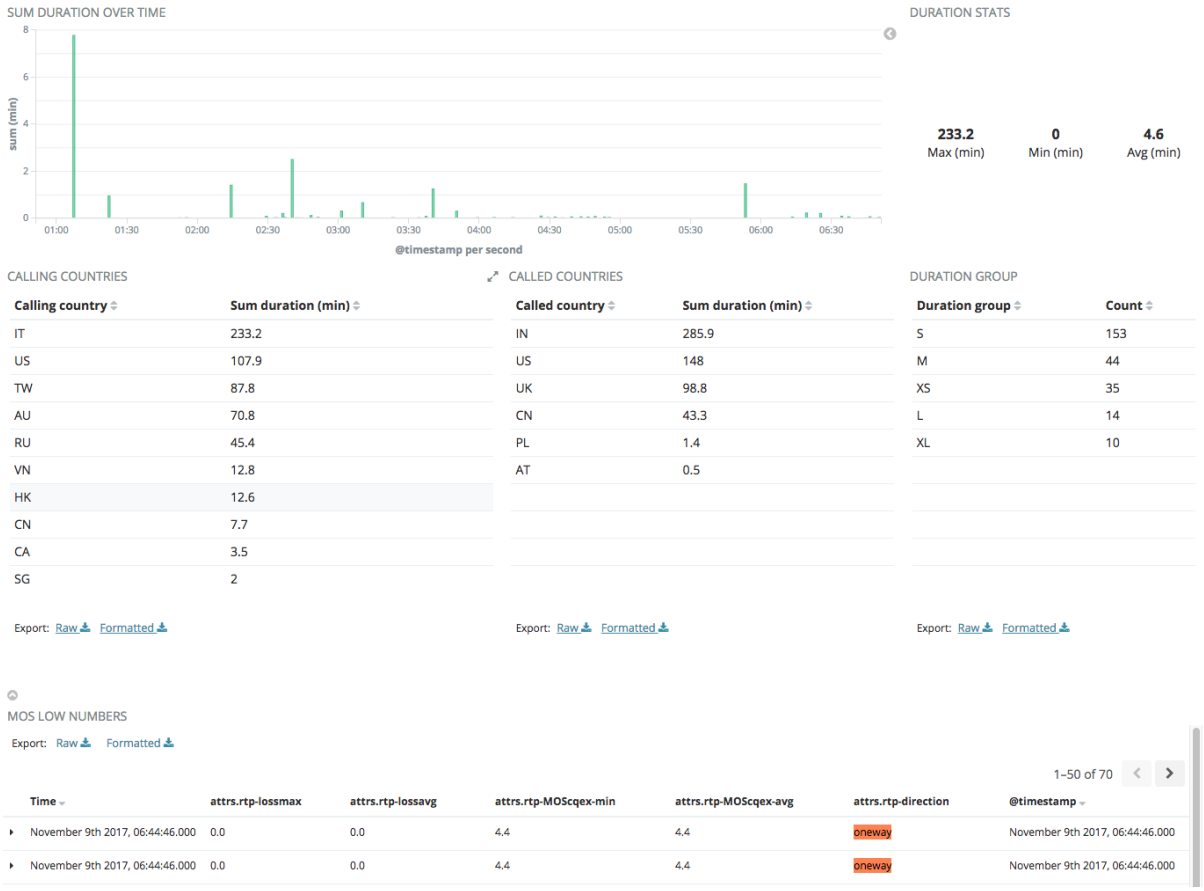


Fig. 26: Screenshot: Lower Part of the Call Dashboard

Note that break-down of calls is calculated differently for source and destination. The source country is determined using the request source address, whereas the destination country is determined from request-URI using knowledge of the SIP service's dialing plan. To accomplish the latter an administrator must tag the calls by a country tag in the ABC SBC. To do so, he must have knowledge of used dialing plans and set the call variable "dst_cc" in ABC SBC rules to proper country codes using the "Set Call Variable" Action, see Figure *ABC Rule for Setting a Destination Country Code by Request URI Prefix*.

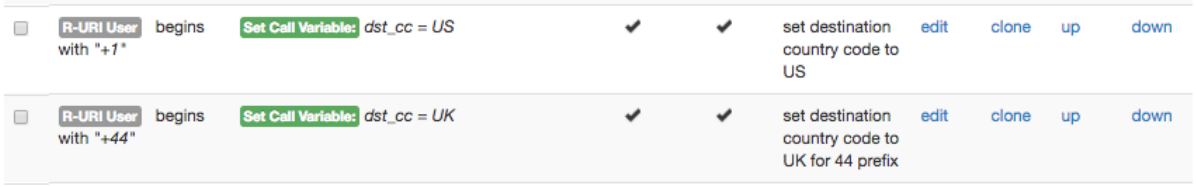


Fig. 27: ABC Rule for Setting a Destination Country Code by Request URI Prefix

The next section highlights calls with suboptimal VoIP quality. Especially of importance are calls with the attribute "attrs.rtp-direction" set to "oneway". That means that for that call, media has been received only in one direction. This is an irritating VoIP phenomena which typically occurs when there are NAT connectivity problems for the affected user.

Last but not least in this dashboard, there is the list of call details. If PCAP and/or WAV recording was enabled for the respective calls in ABC SBC rules, the files or ladder diagrams (see example in Figure *Ladder Diagram for the Suspicious User*) can be downloaded from the unfolded event details. QoS reports are included in the call-stop events in JSON format. The reports include two parts, one for the media streams from and to the caller, and another one for the streams from and to the called party. The values have the following meaning:

- max_delta stands for the maximum interarrival packet gap of received packets. Values above 120 ms for audio are already high and indicate a gap which could have occurred due to muting or voice inactivity detection without marking it as such.
- loss percentage shows relative number of packets lost[2]. Values above 1% show lossy networks, values bellow 5% can be often tolerated by listeners.
- jitter[3] shows variation in packet transit delay. High value above 120 ms typically indicates network congestion and results in dropping of late-arrived packets.

The following example shows such a QoS report. The first bracket pair encloses records about packet streams from caller as seen by ABC SBC and to caller as reported by caller's RTCP reports. The second bracket pair reports quality on streams from and to callee both of them with perfect QoS:


```
[
  {
    "dir": "in",
    "ssrc": "1864183198",
    "src_ip": "192.168.0.155",
    "src_port": "20518",
    "dst_ip": "192.168.0.155",
    "dst_port": "17342",
    "payload": "PCMU/8000",
    "packets": "15670",
    "expected": "16204",
    "bytes": "2695240",
    "last_seq_nr": "42433",
    "max_delta": "14851",
    "max_delta_seq": "41514",
    "gaps": "8",
    "lost_percentage": "3.295482596889657",
    "jitter": "13",
    "dropped": "0",
    "seconds_since_last_received_packet": "0",
    "MOScqex": "3.420"
  },
  {
    "dir": "out",
    "ssrc": "1618416588",
    "src_ip": "192.168.0.155",
    "src_port": "17342",
    "dst_ip": "192.168.0.155",
    "dst_port": "20518",
    "packets": "16271",
    "bytes": "2798612",
    "last_seq_nr": "16280",
    "lost_percentage": "6",
    "jitter": "39",
    "rtt_min": "114",
    "rtt_max": "1150",
    "rtt_avg": "164",
    "seconds_since_last_sent_packet": "0"
  }
]
```


[2] For in-depth discussion of packet loss we recommend the following article: http://www.voiptroubleshooter.com/problems/packetloss.html

[3] For in-depth discussion of jitter and its sources we recommend the following article: http://www.voiptroubleshooter.com/indepth/jittersources.html

8.2.6 Registration Dashboard

The registration dashboard helps to figure out if registration works for SIP users and also identify where they are coming from by analyzing registration events (see Section *Registration Events*). Events reporting on expired registrations are of particular concern because often they mean a user cannot be reached by signaling messages. Most often this is caused by broken home routers, corporate firewalls with a too strict policy, or imperfect SIP client implementations that ignore some important nuances of the SIP RFC3261 contact registration handshake.

The dashboard is structured in several parts shown in Figure *Registration Dashboard*. Bellow the statistic is also a list of the actual registration details (not shown in the Figure).

The first row shows a timeline of the registration events. Our screenshots shows a usual situation in which in every time bucket the number of new registration is about the same as number of deleted and expired registrations. Unusual situations that can be captured here are connectivity outages demonstrated by an increase in expired registrations. Note that SIP devices that register properly and keep re-registering do not produce events as they cause neither a new registration, nor a deleted/expired one.

The second row shows a geographic map of registration events. This gives a rough idea how the users are distributed in the world, even though it is not perfect. That's because the map really shows the events. As mentioned in previous paragraph, not every registered user must be producing registrations events in the examined period of time.

The next row shows use of transport protocols as reported in the registration events, these may include UDP, TCP, TLS and Websockets.

Finally we see the breakdown of SIP User-Agents, here FritzBox being the device producing more registration events, and user accounts that expire most often – probably as result of some NAT traversal difficulties.

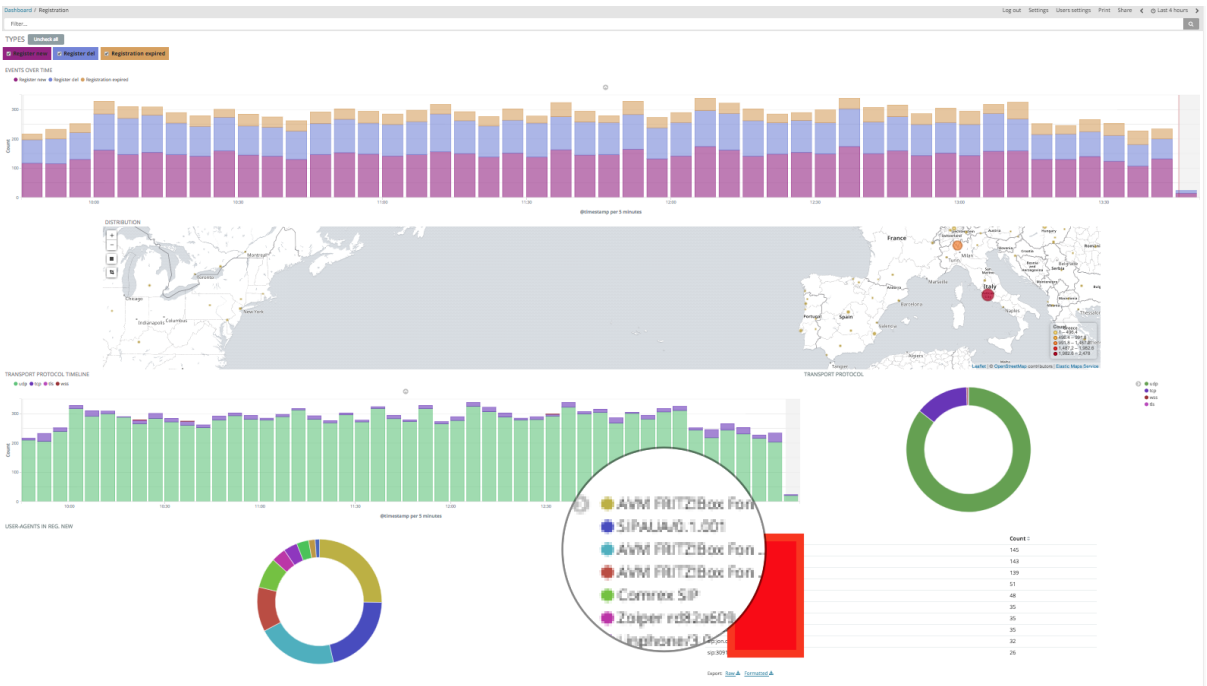


Fig. 28: Registration Dashboard

8.2.7 Connectivity CA Dashboard

This dashboard that came with 4.0 release focuses on topology and visualizes statistics for calls between Call Agents. This helps to discover situations such as a destination Call Agent failing abnormally often to complete calls, or SIP compatibility issues on a link from one CA to another. The numbers visualized in this dashboard refer to the currently chosen time window, as is the case with all other dashboards. The screenshots shown here visualize a situation with five call-agents.

There are two graphs in the top row that provide a quick glance at the situation. The first chart is a directed cyclic relationship graph showing how events, typically call-start, call-attempt and call-end, flow between Call Agents. The thicker the vertices, the more events were generated for traffic on this route. Also the nodes are colored from green to red – the closer to the red side of spectrum the more events refer to traffic from or to the respective node.

The table on the right-hand side shows statistics for calls by destination call agent. This table allows to quickly find out signaling performance of the CA.

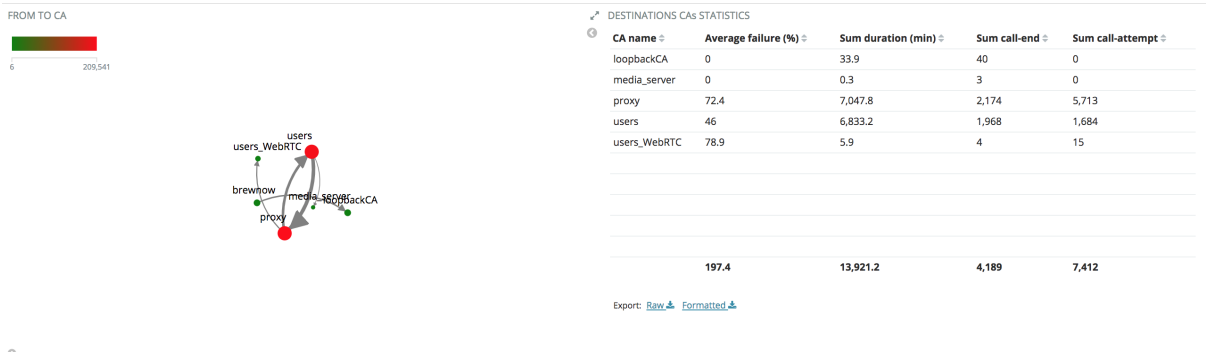


Fig. 29: Top Part of the Connectivity CA Dashboard

In the next rows there are several CAxCA matrixes that visualize the following characteristics of calls between source CA (Y axis) and destination CA (X axis):

- number of call attempts
- connection failure ratio, i.e. number of call attempts divided by sum of call-attempts and call-starts
- duration of calls between CAs
- number of completed calls

Darker colors represent higher numbers, hovering with a mouse over a field shows the actual numbers. In the example screenshot, the darkest failure ratio of 78.9% is shown for the pair proxy->users_WebRTC.

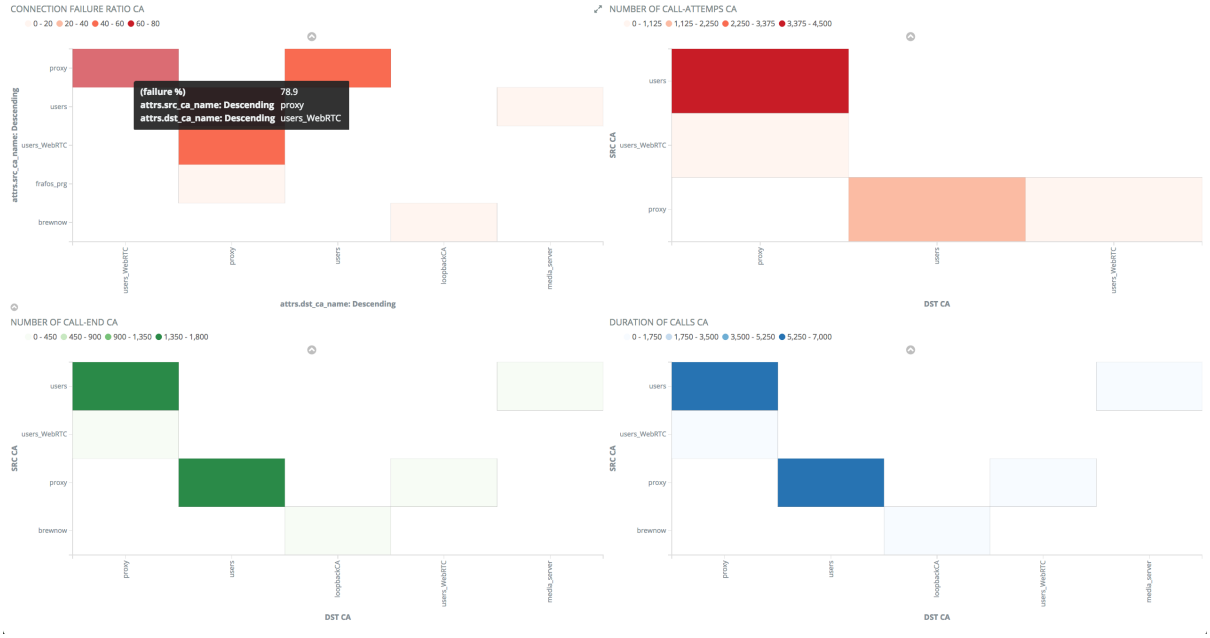


Fig. 30: Bottom Part of the Connectivity CA Dashboard

The CA Connectivity Dashboard can be used for traffic analysis the same way as laid out in the Section *HOWTO Find a Needle in the Haystack: Iterative Event Filtering*. Administrator starts by finding out some aggregated value which appears worth investigating. It can be for example an unusually high failure rate for a SIP connection from Call Agent "proxy" to Call Agent "users_webRTC" as shown in Figure *A CA-CA connection Matrix with High Failure Rate*. This connection shows 78.9% failure rate. It pays off therefore to investigate it in detail.

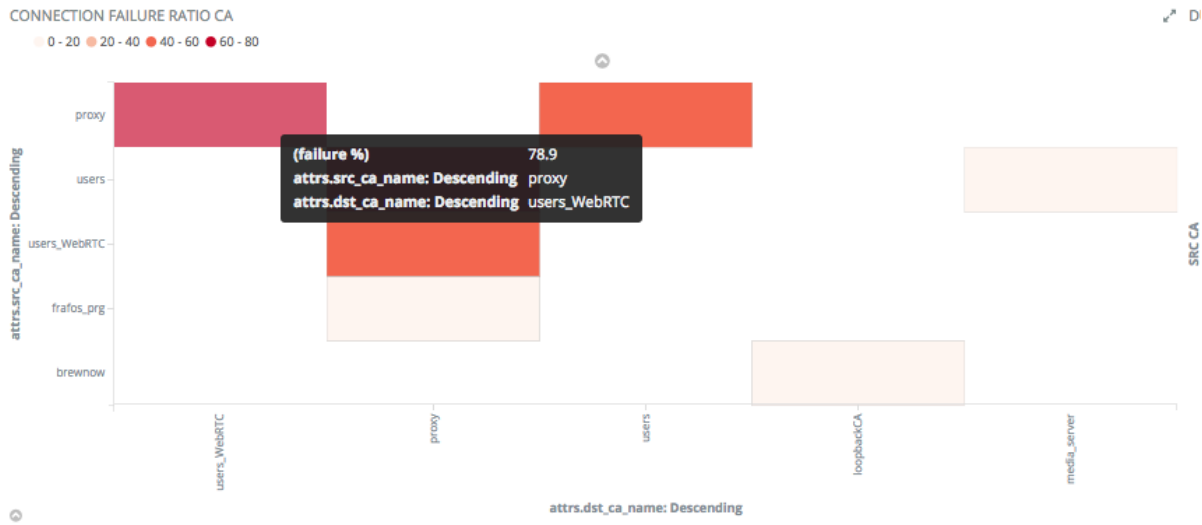


Fig. 31: A CA-CA connection Matrix with High Failure Rate

Narrowing events down to those concerning this connection is as easy as clicking on that particular field in connection matrix and confirm the resulting filter as shown in Figure *Applying a CA-CA filter*.

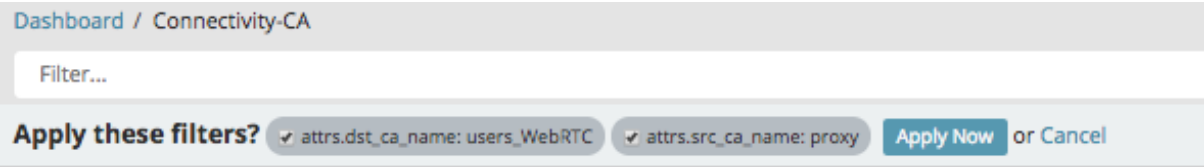


Fig. 32: Applying a CA-CA filter

After applying and pinning the filters, one can switch to the Call Dashboard and inspect the failures for this particular connection in details. Here, one can find that 500 SIP responses dominate and inspect the details of the respective events.

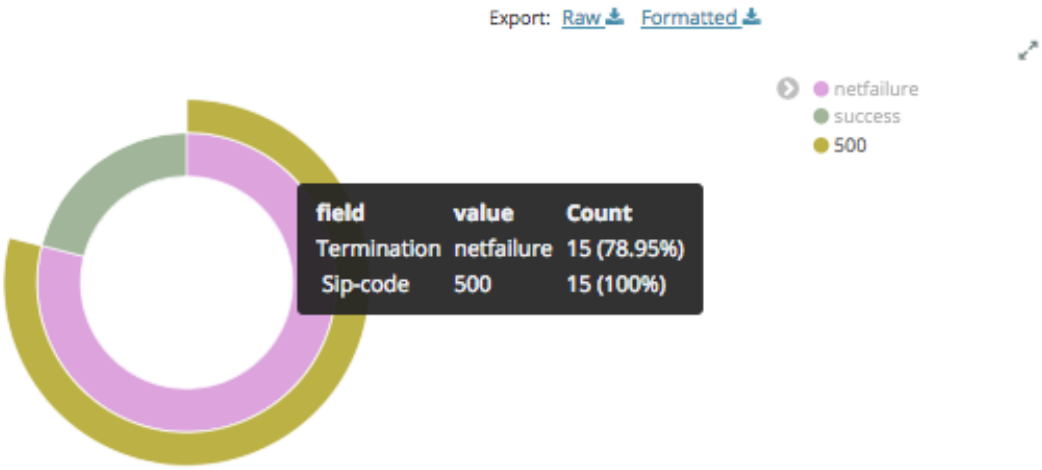


Fig. 33: Finding out the Root Cause of High CA-CA Failure rate

The bottom-most Connectivity Dashboard lane shows availability of the monitored Call Agents. Only Call Agents for which monitoring has been enabled are shown (See Section *IP Blacklisting: Adaptive Availability Management*). The 0 status represents a Call Agent that is reachable, all other values represent some kind of connectivity issues (unreachable, DNS-unresolvable, overloaded or returning a negative response).

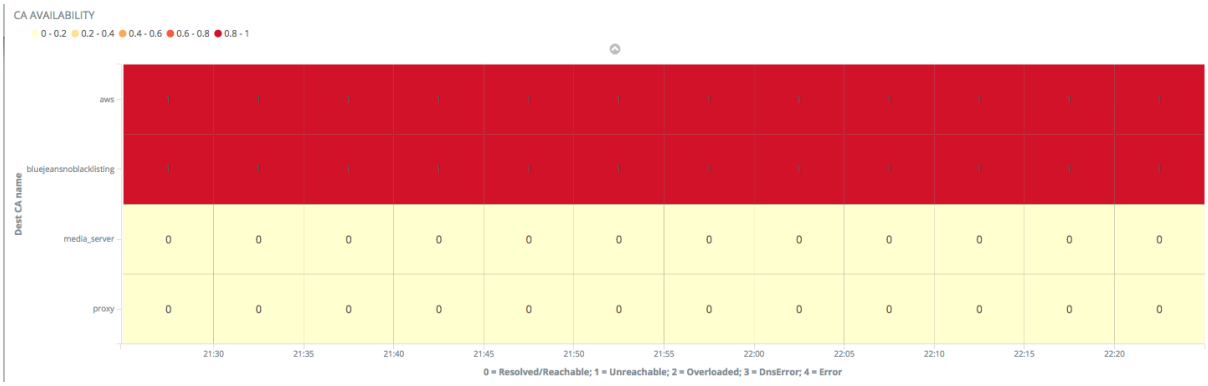


Fig. 34: Call Agent Availability Lanes

8.2.8 Security Dashboard

Security Dashboard is perhaps one of the most important ones as it tracks events relating to security as explained in Section *Security Events* and attempts to answer the question where the attacks are coming from. Occurrence of such events may indicate an attack that can compromise security of a SIP user or of the whole service. A detailed debate of security techniques recommended to fortify a SIP service against attackers is provided in Section *Securing SIP Networks using ABC SBC and ABC Monitor (optional)*.

The security dashboard comes with three important charts in the Toplist section: the most frequent offenders by originating IP address, /24 netmask and geographic region.

Unless an attacker is mounting a sophisticated distributed attack, the top-list shows which IP address is causing the most of offending traffic. It is as easy as a single click of button to limit all events to those caused by the offending IP address, inspect these and undertake some appropriate security measures, blocking the IP address typically. Even if some more sophisticated attackers can send small batches of traffic from multiple IP addresses in the same subnet – they will appear on the /24 subnet toplist.

The geographic map is also very important from the security point of view. Even though most attackers don't avail of many IP addresses, they sometimes do use more than one subnet to stay under radar screen. As long as they do not use VPNs, these can be tied down by their geographic region.

An example of a situation in a public SIP service is shown in Figure *Example of the Security Dashboard*. It shows the most active IP addresses violating the SIP site's policies, and also their break-down by subnet and country, China being the most active source of offending traffic.

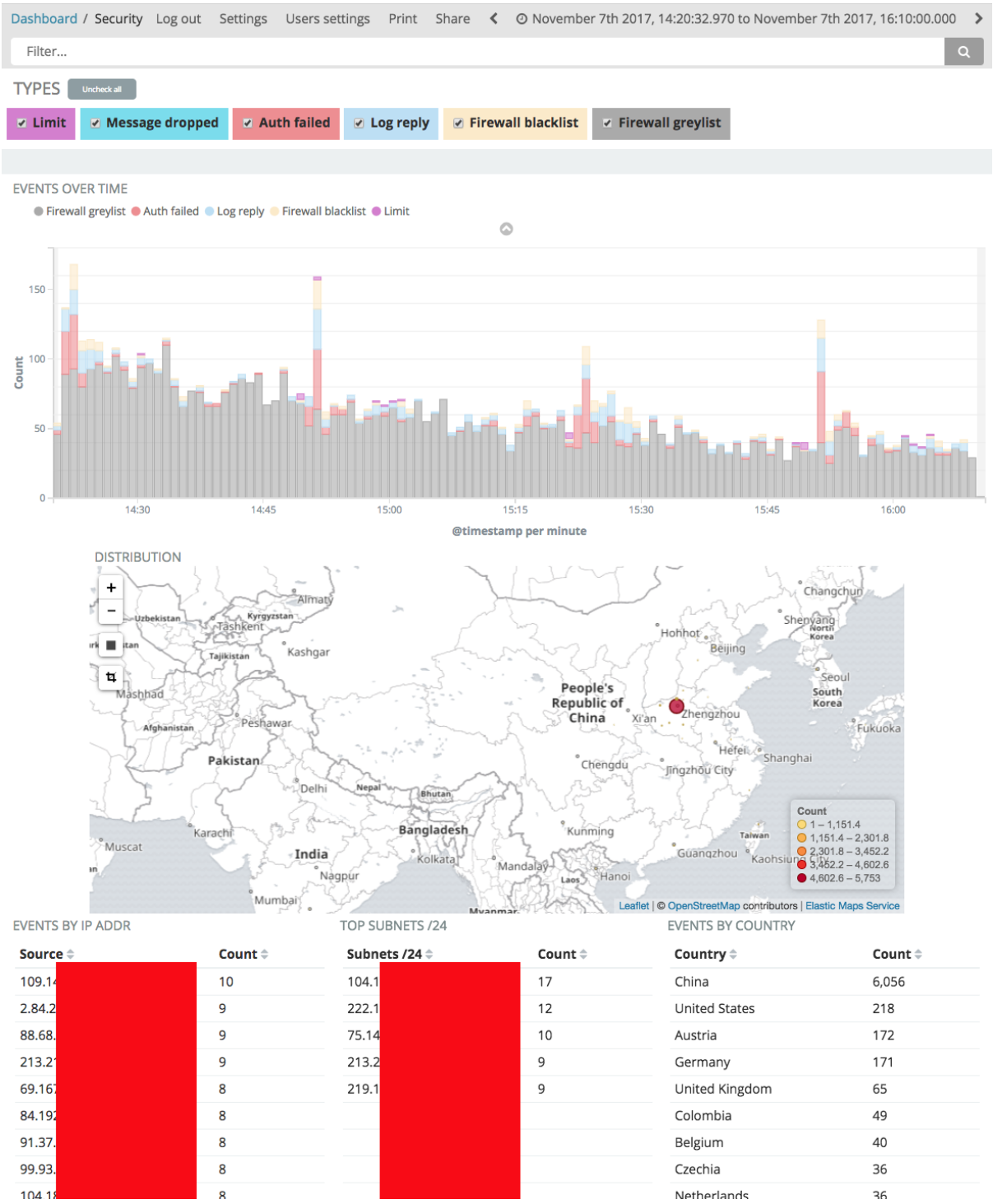


Fig. 35: Example of the Security Dashboard

8.2.9 Exceeded Limits Dashboards

Trying to find some unusual patterns may be sometimes a repetitive task. Therefore it is possible to raise alerts when some abnormal conditions repeat too often. The ABC Monitor allows to configure such alerts under "Settings" and inspect the alerts in the Exceeded Limits Dashboard. There are several types of the alerts, that are described in subsequent subsections.

The dashboard is only of advisory nature: it highlights excessive traffic but does not take a further action. Administrator must act if he chooses to. The following example in Figure *Exceeded Limits Dashboard* shows various such alerts as they occur over time. The donut chart breaks down the number of alerts by their type, the most offending URIs are shown in the top-chart on the right-hand side.

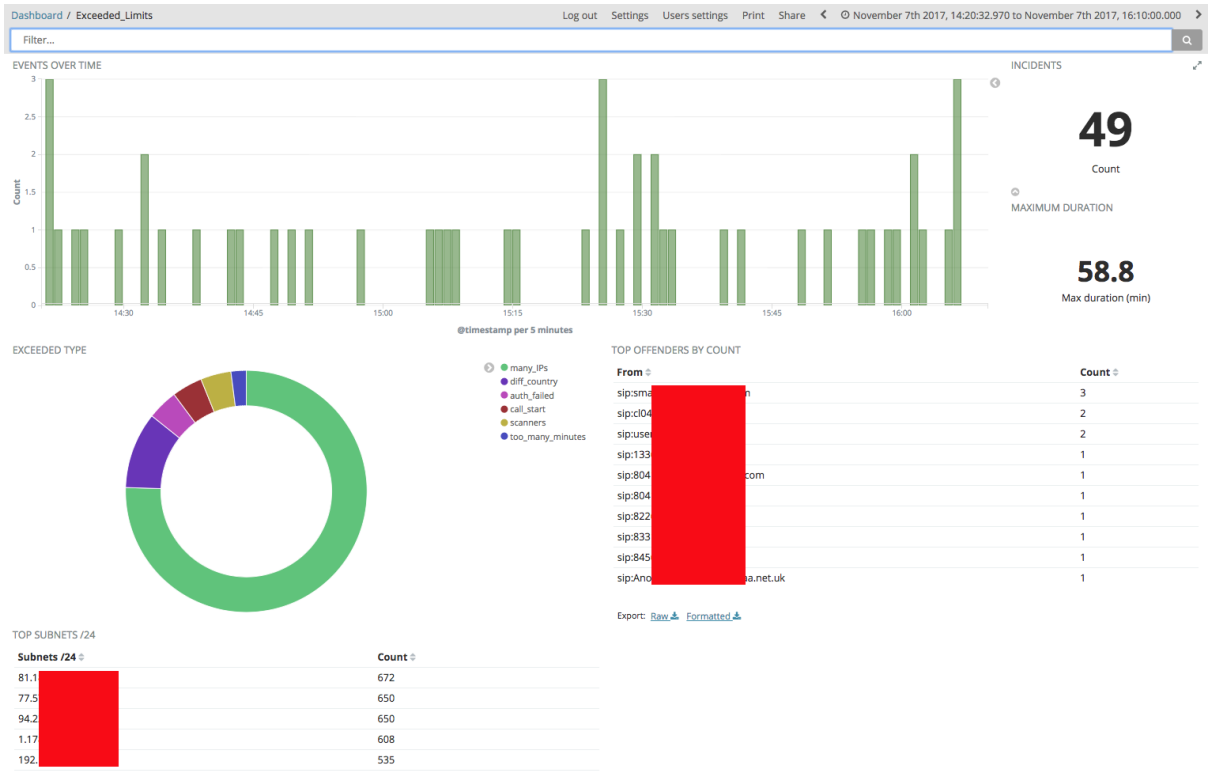


Fig. 36: Exceeded Limits Dashboard

Threshold for the respective alert types can be configured from the **Settings** Menu. The actual values can vary from site to site as what is legitimate for a SIP service may be alarming for another service. For example, in a domestic VoIP service reporting a change of user's country may be worth inspecting, whereas the same report is usual in a global public SIP service.

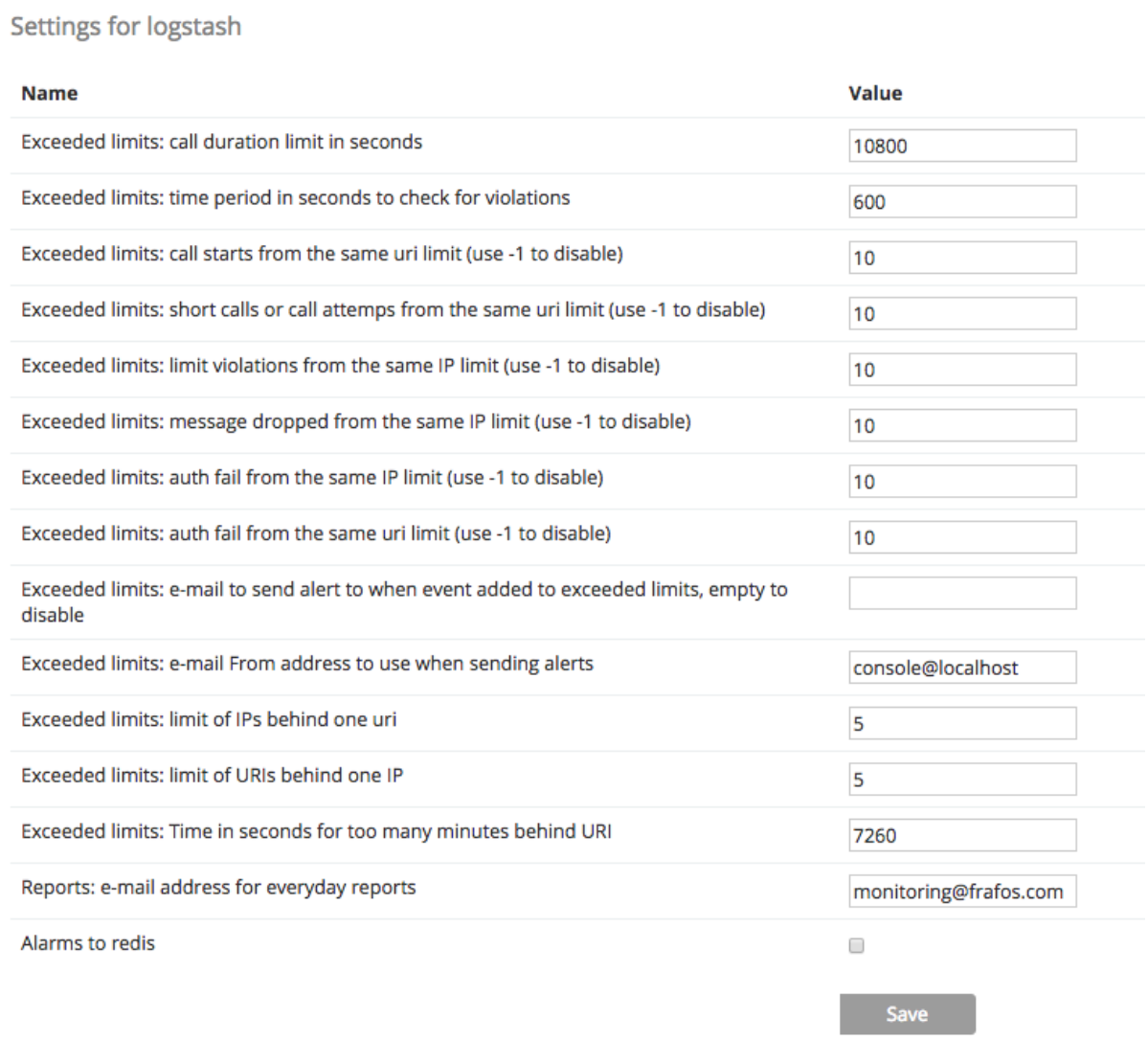


Fig. 37: Alert Threshold Settings

Maximum Call Duration (max_duration)

This alert is raised when a call is completed that exceeded a maximum call length threshold. The threshold value is configurable under “Settings”. Default value is 10800 seconds (3 hours).

Too Frequent Call Attempts from a URI (call_start)

This alert is raised when a user identified by his URI makes too many call attempts. This can be caused for example by a SIP scanner. The number of attempts and the time-span are configurable under settings and default to 10 attempts for previous 10 minutes.

Too Frequent Call Attempts or Short Calls from a URI (scanners)

This is similar to the previous alert except very short calls bellow 0.5 seconds count towards the limit as well.

Repeated Traffic Shaping Violations from an IP (limit)

This alert is raised when too many limit events originate from a single IP address over a period of time. By default, 10 such limit event occurrences over past 10 minutes will raise the alert.

Repeated Drop for an IP Address (message-drop)

This alert is raised when the rule action **drop** in an SBC drops an incoming SIP request from an IP address too often, by default 10 times in the past 10 minutes.

Too Many Authentication Failures from an IP Address (auth_failed)

This alert is raised when an authentication fails too many times from a single IP address. By default, 10 attempts in past 10 minutes from the same IP address will raise the alert.

Too Many Authentication Failures from a URI (auth_failed)

This alert is raised when an authentication fails too many times from a single URI. By default, 10 attempts in past 10 minutes from the same IP address will raise the alert.

Rapid Growth in Number of Security Events (security_metrics)

This alert is raised when the number of security events (drop, limit, auth-failed, log-reply) begins to grow too quickly.

A URI Active from behind too many IP addresses (many_IPs)

This alert allows to detect situations in which a user as identified by his From URI is spotted at too many IP addresses. This may be caused by both legitimate and illegitimate behaviour. Sometimes users like to be reachable under the same URI at multiple destinations (office, home, second-home) or multiple call agents may be registered under the same call center's SIP AoR. However it may be also a case of identity theft. The alert includes number of IPs found, and the actual IP addresses if there are fewer than five of them. The alert doesn't repeat until next day.

Too Many Users behind a single IP Address (many_URIs)

This alert is triggered when events from too many users appear coming from a single IP address. This may be often legitimate when there are multiple users behind a home NAT, carrier NAT, or a PBX. The URI count is shown in the alert (countURI field) and so are the actual URIs if there are fewer than five of them (URIs field). The alert doesn't repeat until next day.

Changed Country Alert (diff_country)

This alert is raised when a new registration, call attempt or call with the same From URI comes from a different country than previously in the past 24 hours. The alert may identify both legitimate cases (users or call-centers with presence in multiple countries) as well as identity theft. The field **firstCountry** shows the country that was encountered previously, **geoip.country_name** show the current event's country name.

Too Many Minutes from a User (too_many_minutes)

This alert is raised for tagged calls when a user identified by his From URI address makes too many call minutes in the observed period of time. By default 7260 minutes in the past two hours will trigger the alert. The field "durationSum" shows the offending number of seconds.

To tag calls to count against the many-minutes alert, set the call variable **minute_counter** to the value **enabled**. This can be particularly useful in topologies when a call on a way to and from a PBX passes the SBC more than once, see Figure *Setting Minute Counter Call Variable*.

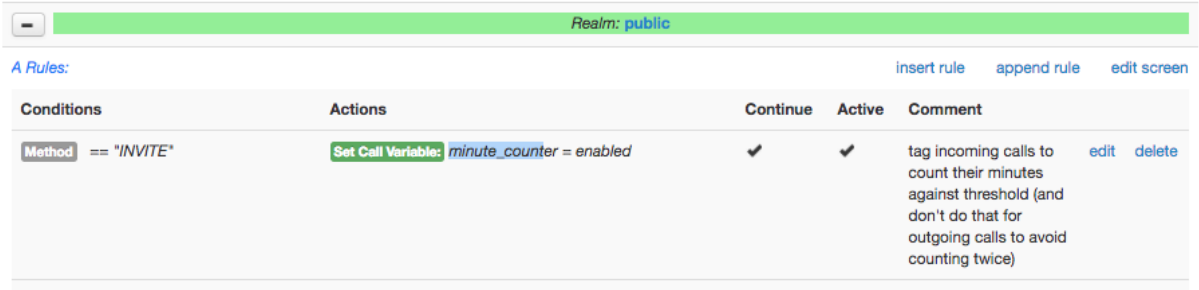


Fig. 38: Setting Minute Counter Call Variable

Underperforming Destination Call Agent (poor_failure_ratio_ca)

This alert is raised when the number of call attempts is relatively high to the number of successful calls. This may often be the case when a destination Call Agent begins to be overloaded. The alert is raised when the failure ratio exceeds 90%.

8.2.10 System Dashboard

The system dashboard shows utilization of the ABC SBC linux operating system: system load, memory and CPU.

The example screenshot shown in Figure *System Dashboard* shows utilization of an SBC. The situation here is normal as all the values keep oscillating within a fixed range.

Fig. 39: System Dashboard

8.2.11 Network and Statistics Dashboard

The network statistics dashboard shows amount of traffic processed by all of the managed SBCs, both at high-level (number of calls and registrations) and low-level (number of bytes and packets). It also shows statistics of automated blacklisting.

The example in the Figure *Network Statistics Dashboard Capturing a Failover Situation* shows a typical situation on a public SIP service. The number of registrations remains fixed over time at about 3 thousands. Parallel calls peak at 8 PM, and a moderate number of auto-blacklisted IP addressed reaches slightly above one hundred. The number of greylisted IP addresses is quite high though: at 70,000 and constantly increasing. That's a clear evidence that the public SIP service is continuously subject to SIP scanning. The number of IP addresses that have passed greylisting is slightly higher than number of registrations: obviously some registrations and re-registrations occur in about same quantity, leaving the number of current registrations constant, and increasing number of IP addresses that have been accepted over time.

Fig. 40: Network Statistics Dashboard Capturing a Failover Situation

Particularly the number of calls and registrations is important – a dip often almost always indicates some abnormal network conditions. For example, if the SIP services loses its IP connectivity, SIP re-registrations will not reach it and subsequently the number of current registrations sinks down.

8.2.12 Diagnostics Dashboard

The diagnostics dashboard collects details that help with troubleshooting of low-level problems. Such may include SIP and SDP interoperability problems, or QoS problems. The dashboard shows events described in Section *Diagnostics Events*. ABC Monitor visualizes these events in several dashboards: Diagnostics, Transport and Connectivity CA.

Most of these events appear only when activated by administrator in the ABC SBC rules. The key diagnostics feature the ability to store PCAP files of SIP/RTP and WAV audio files. Being able to retrospectively inspect these allow administrators to find a problem which appears only transiently and is hard to reproduce.

In order for these files to appear in the dashboard, the ABC SBC must be configured to produce them. Once configured for selected calls, as soon as they complete, ABC SBC uploads the resulting WAV and PCAP files to ABC Monitor. Eventually administrator can download them from the diagnostics dashboard by clicking on the respective event details. There is also a possibility to see the SIP traffic in from of a ladder chart as shown in Figure *Ladder Diagram for the Suspicious User*.

It is worth noting that this ABC SBC capability to report on the traffic as seen "from inside" is superior to the capability of external snooping-based monitoring equipment. The "insider view" allows to analyze such SIP traffic even when it is encrypted when on the net, or obfuscated using Topology Hiding (see Section *Topology Hiding*.)

To activate recoding of the SIP traffic one must use the action “Log received traffic”. When this action is called for a SIP call, the SIP signaling is recorded in PCAP file, optionally including RTP traffic.

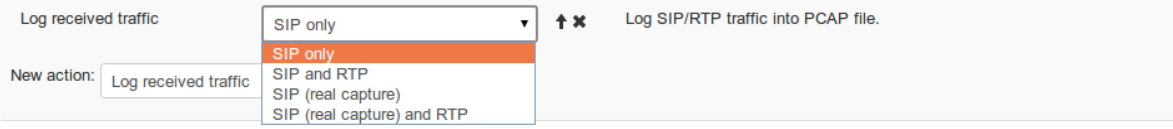


Fig. 41: Configuring traffic capturing

When recording completes, a “message-log” event is produced that includes a references to the stored PCAP file.

The parameter “PCAP file name” allows administrators to define their own filename for the PCAP files. Using Replacement expressions (see *Using Replacements in Rules*) one can include SIP message elements in the filename that may make identification and sorting the recorded files easier. If no filename is chosen, the ABC SBC chooses its own ephemeral filenames. A fixed name may result in mixed PCAPs for different transactions. If custom filename is being used, it is recommended to not use fixed filename but include some date or time variable replacement in the filename to make it unique. In any case, the filename is relative to the path /data/traffic_log to avoid conflicts with the filesystem. Use a filename with .pcap extension.

Note that this action cannot be used multiple times for the same call meaningfully. In such case an error is reported in the SBC process log and only the first used logging action takes effect. Also due to the “inside view” nature how the packets are captured, they may display some minor differences from the actual traffic as seen on the net. Specifically TCP headers are not shown for SIP traffic sent using TCP.

Custom events also appear in this dashboard and also require a proper configuration on the SBC side as shown in the Section *Diagnostics Events*.

8.2.13 Monitor Troubleshooting

Should the ABC Monitor itself become a bottleneck in a network, it is a good idea to check its status. To see the status page, open its URL with path “/status” as shown in the screenshot Figure *Displaying ABC Monitor Status*. If the status shown on the page is not “green”, collect the statistics and contact FRAFOS support.

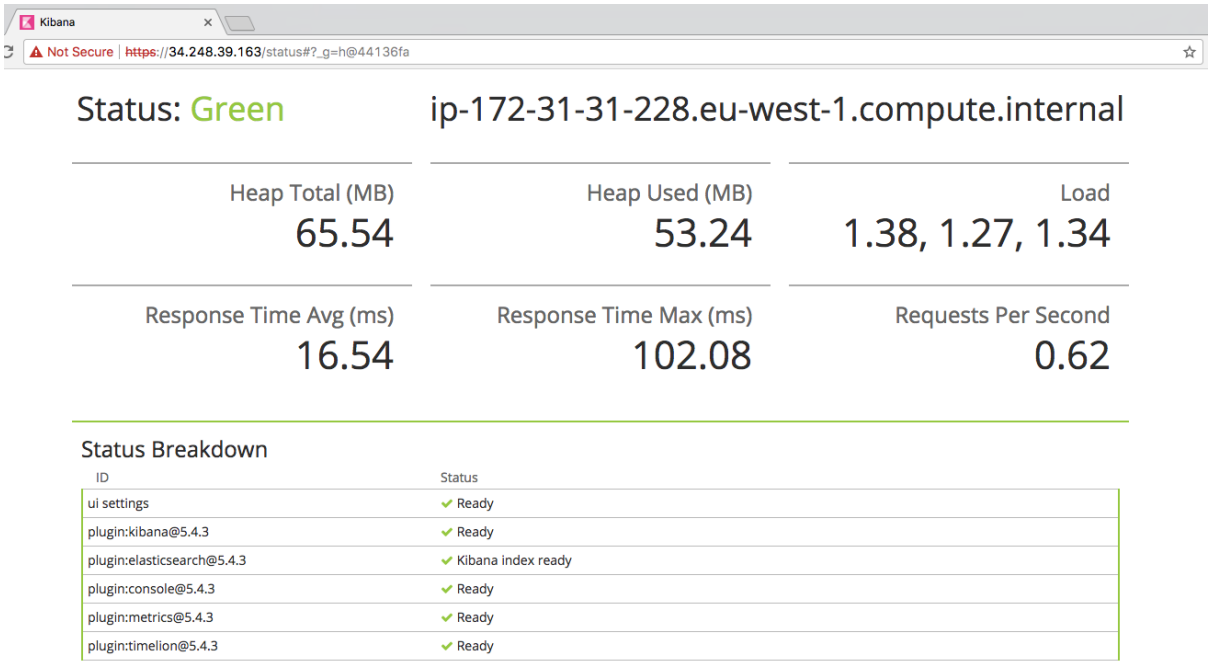


Fig. 42: Displaying ABC Monitor Status

8.3 Live ABC SBC Information

The Frafos ABC SBC allows to inspect its internal state in the administrative GUI.

8.3.1 Registration Cache

Registration Cache plays a significant role in off-loading registers, see Section *Registration Caching and Handling* for more details. The actual Content of the ABC SBC registration cache can be inspected using the web interface under the **"Monitoring → Registration cache"** link, see Fig. *Registration cache*.

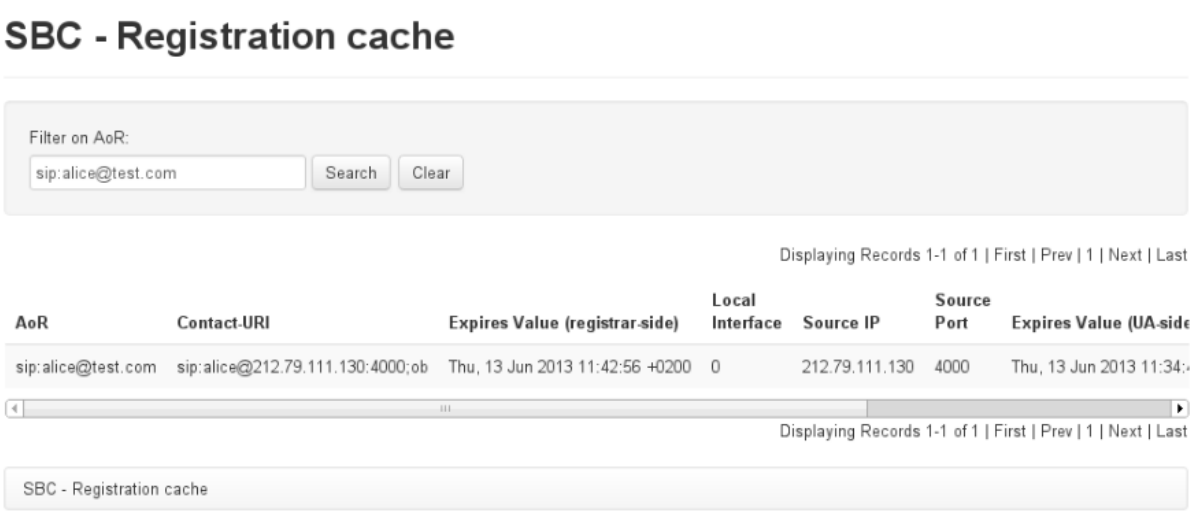


Fig. 43: Registration cache

The following information is displayed for each entry:

- *AoR* - Address of Record. SIP URI address that is associated with none, one or more user Contacts by the SIP registration procedure.
- *Contact-URI* - Contact registered by the user agent and associated with an AoR.
- *Expires Value (registrar-side)* - registration expiration at registrar side. This is the time when both the downstream registrar and the ABC SBC will let the contact expire.
- *Expires Value (UA-side)* - registration expiration at client (UA). This is the time when the ABC SBC expects the client to re-register. Failures to re-register timely are ignored to keep the client reachable even if it its re-registration procedure doesn't work accurately. Because of REGISTER throttling feature (see Section *Registrar off-load*) the actual value may be different (earlier) from *Expires Value at registrar-side*.
- *Source IP* - IP address where the REGISTER was received from
- *Source Port* - port where the REGISTER was received from
- *User Agent* - user agent identity (content of User-Agent header in REGISTER message)

8.3.2 Live Calls

"Monitoring → Live calls" shows list of active calls, i.e. calls that have been forwarded and established. The calls appear there from the time when a 200 SIP response is received from a downstream SIP element, till the call is terminated. Calls that are in so-called "early media" or "ringing" status do not show, neither are locally processed calls shown (e.g. calls processed using *Onboard Conferencing*).

Since the ABC SBC acts as a SIP B2B user agent, two call legs are shown for each established call:

- A leg (originating leg) - SIP dialog established with caller
- B leg (terminating leg) - SIP dialog established with callee.

Information displayed for each call leg include:

- Source IP - IP address where the REGISTER was received from
- Source Port - port where the REGISTER was received from
- Call-id - SIP dialog identifier
- Remote party - URI of remote party (equals the *From* URI for A leg and the *To* URI in case of B leg)
- Remote target -Contact of remote party
- Local party - Local URI.
- Dialog state - Current state of the SIP dialog
- Call start time - Time of call setup

The administrator can manually terminate the call using the **"kill"** link and inspect call status details using the **"Call Status Information"** link.

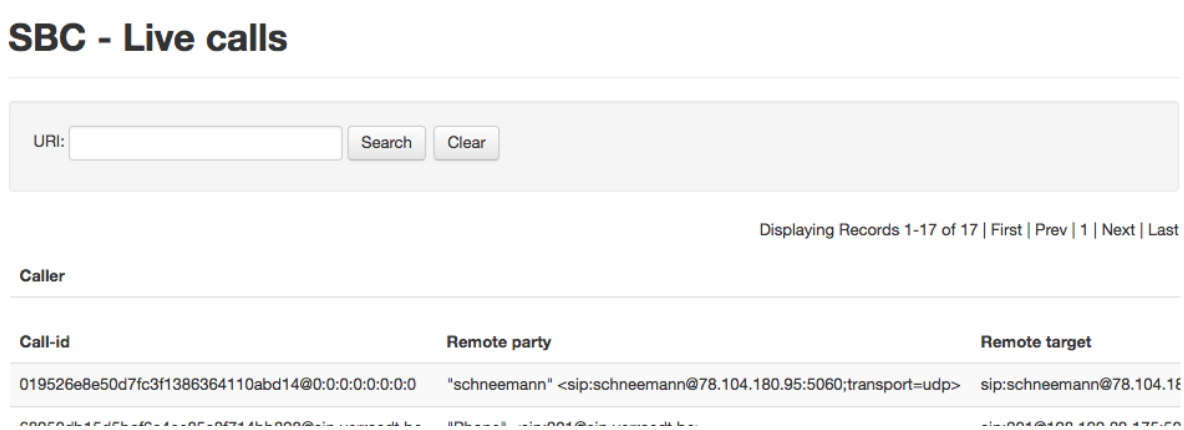


Fig. 44: Live Calls

8.3.3 Destination Blacklists

"Monitoring → Destination Blacklists" shows IP addresses that have been found to be unresponsive. See section *IP Blacklisting: Adaptive Availability Management* to find out how to configure the ABC SBC to handle routing to unresponsive SIP destinations.

The Figure *Destination Blacklists* shows the user-interface for monitoring the unavailable IP addresses. It shows a single IP address and time-to-live to remain on the availability blacklist.

The TTL field specify the time interval (in seconds) for which the destination is put on blacklist. When this interval pass, the destination is automatically removed from the blacklist. If '-1' value is used for TTL or if "Valid forever" checkbox is checked the destination is put on blacklist forever or until it is removed manually.

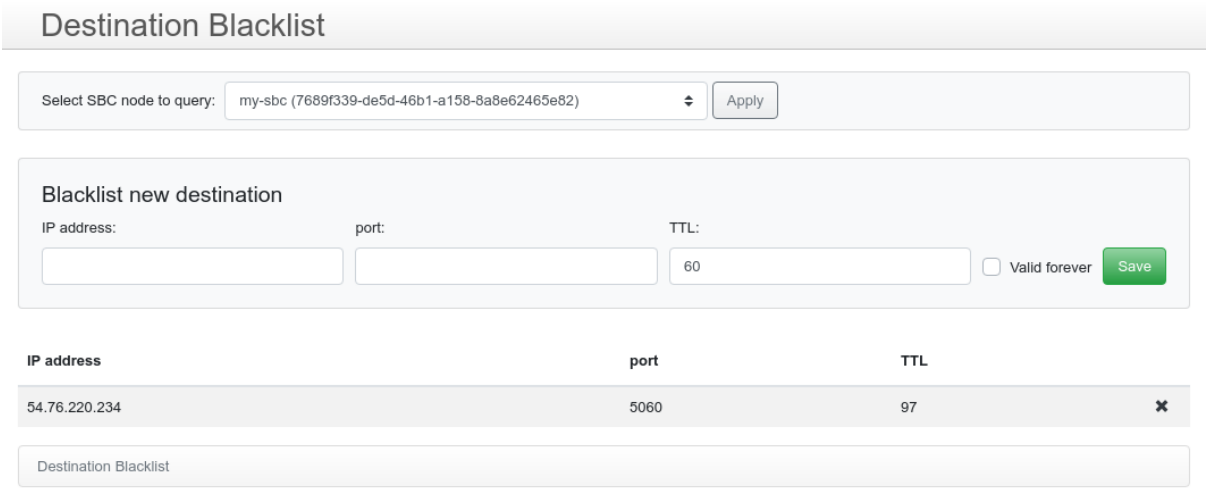


Fig. 45: Destination Blacklists

8.3.4 User Recent Traffic

The ABC SBC always keeps track of the most recent SIP traffic. This is particularly useful when a problem is identified which doesn't occur anymore and needs to be troubleshooted retro-actively. Another reason to look-back in this stored traffic is it includes even IP packets that are filtered at IP layer (see Section *Police: Devising Security Rules in the ABC SBC*) and cannot be troubleshooted at higher layers.

By default, there are 10 files of max. 50 MB size rotated to store the captured traffic. The file size and number of files to keep should be tuned according to available disk space. It can be configured or disabled using global config options under "Config → Global Config → Pcap" page.

Note: the files use extensions ".pcapXX", where the "XX" part corresponds to the file number. If the global config option to set number of files to keep is lowered, the older files (those ones that are not going to be overwritten by next cycle) are kept for the case the data is still needed, and have to be deleted by admin.

The administrative page "Monitoring → User Recent Traffic" allows administrators to retrieve SIP traffic for a specific IP address. Also a secondary IP address can be included in which case packets matching either IP address will be retrieved. The retention policy for the stored traffic can be configured as shown in the Section *PCAP Parameters*.

To retrieve the SIP traffic, a user must choose the time interval within the available retention period (configuration of which is described in Section *PCAP Parameters*), IP address, and press the "Get PCAP file" button. Processing can take up to several minutes depending on the time interval chosen. The traffic comes in an archive in PCAP format along with TLS session keys that can be used to decrypt SIP traffic that came over TLS connections.

Wireshark can be used to inspect encrypted TLS traffic using the session keys, see the following link for a detailed HOWTO: https://jimshaver.net/2015/02/11/decrypting-tls-browser-traffic-with-wireshark-the-easy-way/

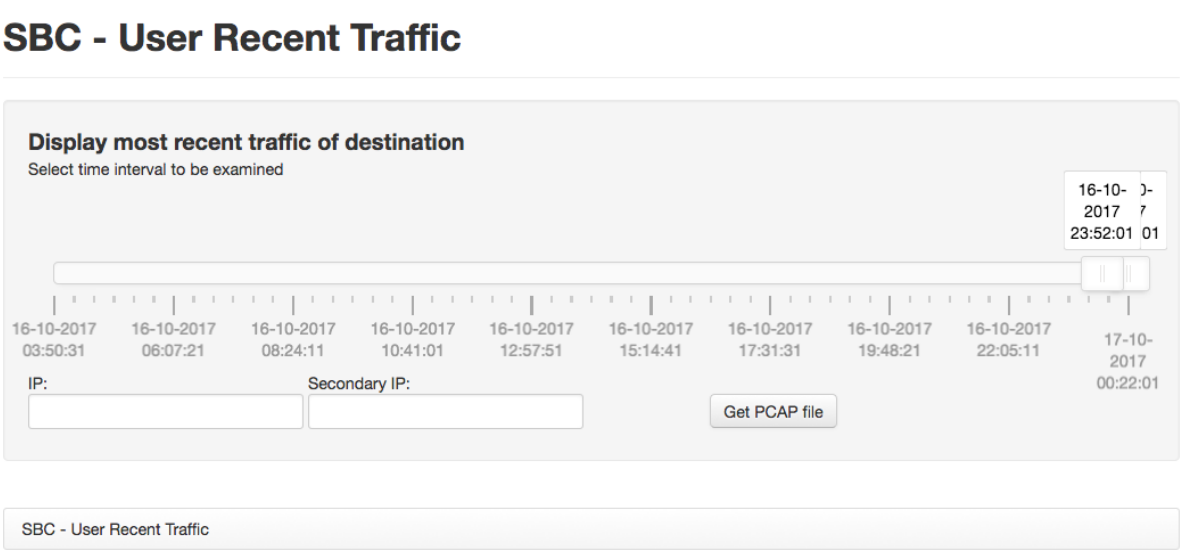


Fig. 46: User Recent Traffic

8.4 Using SNMP for Measurements and Monitoring

The SBC provides SIP and RTP traffic related counters. These measurements are exposed to external monitoring tools using SNMP API. The administrator can also manually use standard SNMP tools (e.g. "snmptable" or "snmpwalk" commands).

The SNMP daemon uses the external management interface (XMI) and is configured in the **"Config → Global Config → SNMP"** screen. The ABC SBC collects general, per Realm/Call Agent and user defined measurements. Complete SBC counters specification in MIB format is available in the *" /usr/share/snmp/mibs/FRAFOS-STATS-MIB.txt"* file.

8.4.1 General Statistics

General statistics present the number of the calls currently processed by the system.

- **fSBCCalls** - number of active calls
- **fSBCCallStarts** - number of call attempts
- **fSBCBits** - RTP Bits relayed
- **fSBCRegs** - number of SIP registrations

The following example shows the current number of calls. Note that on your system you must use its administrative IP address instead of the address shown in the example, the SNMP port configured under SNMP app (if configured differently from the default 161), and that the -c parameter must be set to the current SNMP community value if changed under "Config → Global Config → SNMP":

```
% snmpwalk  -v 2c -c sbc_com_321 172.31.2.42 FRAFOS-STATS-MIB::fSBCCalls
FRAFOS-STATS-MIB::fSBCCalls.0 = INTEGER: 1
```

8.4.2 Statistics per Realm / Call Agent

These measurements are counted for each Realm and Call Agent separately.

- **UUID** - unique identifier
- **Name** - Realm resp. Call Agent name
- **RealmName** - Realm name which a Call Agent belongs to (shown for call agent only)
- **CallsStartsTo** - number of call attempts to the Realm/Call Agent
- **CallStartsFrom** - number of call attempts from the Realm/Call Agent
- **CallsTo** - number of call attempts to the Realm/Call Agent (including calls in progress)
- **CallsFrom** - number of call attempts to the Realm/Call Agent (including calls in progress)
- **BitsTo** - RTP Bits relayed to the Realm/Call Agent
- **BitsFrom** - RTP Bits relayed from the Realm/Call Agent

The following example provides a snapshot of statistics collected by the ABC SBC using the *snmptable* command:

```
% snmptable -v 2c -Cb -CB -c sbc_com_321 172.31.2.42 \
-Oqq FRAFOS-STATS-MIB::fSBCRealmStatsTable
SNMP table: FRAFOS-STATS-MIB::fSBCRealmStatsTable
                                    UUID      Name CallStartsTo CallStartsFrom CallsTo␣
↪CallsFrom BitsTo BitsFrom
1356fb76-290c-cc49-4b46-00007784bfc6 sip-realm            2              0       2␣
↪         0  42656    51690
5fa54bf5-01d5-56e9-23b4-000019b29424 rtc-realm            0              2       0␣
↪         1  51690    42656
```

8.4.3 Call Agent destination status

These measurement are exported from the destination monitor.

- **UUID** - status uuid (call agent uuid + resolved)
- **Realm** - realm name
- **CaName** - call agent name
- **Dest** - raw destination address
- **Resolved** - resolved destination address
- **Status** - destination status
- **BlTTL** - black list time to live

The following example provides a snapshot of statistics collected by the ABC SBC using the *snmptable* command:

```
% snmptable -v 2c -Cb -CB -c sbc_com_321 172.31.2.42 \
-Oqq FRAFOS-STATS-MIB::fSBCCADestStatusTable
SNMP table: FRAFOS-STATS-MIB::fSBCCaDestStatusTable
                             UUID    Realm      CAName                  Dest       ␣
↪Resolved      Status BLTTL
bbece1ad7e9e89224f82d57b10de6feb default testing_two proxy.frafostest.net sip:10.0.
↪1.111 Unreachable   100
1a8290e08c9e53623548c5695e469340 default testing_two proxy.frafostest.net sip:10.0.
↪1.119 Unreachable   100
3e70cb3020b0bcd3923311bdb000950a default testing_two proxy.frafostest.net sip:10.0.
↪1.118 Unreachable   100
ced88da8867807accd7c20b4ec21695a    test     testing proxy.frafostest.net sip:10.0.
↪1.119 Unreachable   100
```

(continues on next page)

(continued from previous page)

```
c6c1db2869f403303c0543d733d38f8e    test     testing proxy.frafostest.net sip:10.0.
↪1.111 Unreachable   100
cf3869ccba76acb5ca611691f1b0b347    test     testing proxy.frafostest.net sip:10.0.
↪1.118 Unreachable   100
```

8.4.4 Interfaces statistic

These measurement are exported from the transport layer.

- **Name** - interface name
- **ReqSent** - number of sent requests
- **RepSent** - number of sent replies
- **ReqRecv** - number of received requests
- **RepRecv** - number of received replies
- **ReqSentRetr** - number of sent retransmitted requests
- **RepSentRetr** - number of sent retransmitted replies

The following example provides a snapshot of statistics collected by the ABC SBC using the *snmptable* command:

```
% snmptable -v 2c -Cb -CB -c sbc_com_321 192.168.8.134 \
-Oqq FRAFOS-STATS-MIB::fSBCInterfaceStatsTable
SNMP table: FRAFOS-STATS-MIB::fSBCInterfaceStatsTable

Name ReqSent RepSent ReqRecv RepRecv ReqSentRetr RepSentRetr
sig         6       8       6       6           0           0
```

8.4.5 User Defined Counters

User defined counters can be created and increased using an **"Increment SNMP counter"** action configured in inbound or outbound rules. This action increments a user-defined SNMP counter by a given value. As parameters the counter name and the counter increment are given, see Fig. *User defined counters*.

Fig. 47: User defined counters

The value of the custom counters can be queried using the *snmptable* command:

```
% snmptable -v 2c -Cb -CB -c sbc_com_321 public 172.31.2.42 \
-Oqq FRAFOS-STATS-MIB::fSBCCustStatsTable
FRAFOS-STATS-MIB::fSBCCustStatsTable.1.2.1 "gui.alice_calls"
FRAFOS-STATS-MIB::fSBCCustStatsTable.1.3.1 12
```

8.4.6 SNMP traps

The SNMP daemon can generate SNMP traps (alerts). This functionality is disabled by default and can be enabled in **"Config → Global Config → SNMP"** screen by entering the trap receiver (manager) address. The trap receiver shall be entered in format *"HOST [COMMUNITY [PORT]]"*. The generated traps can use SNMP protocol v1 or v2c (or both, but do not send both to the same receiver). The time interval between checks and sending the SNMP traps is 10 minutes. The SNMP traps are sent when any of the following conditions is met, and only if the check state changes since last check:

- system interface link goes down or up
- disk free space drops below 10%
- system load gets over 15 (1min average) or 10 (5min average) or 5 (15min average)

8.5 Command-line SBC Process Management

Occasionally it may be useful to review or change the low-level status of daemons that implement the SBC functionality. ABC SBC is running several daemons for processing and controlling signalling and media traffic, management services like a web interface, configuration management, and others.

To control all these processes, the systemd daemon is running:

- systemd (Section *Process Management using Systemd*)

There are two actual work-horses: the signaling and database daemons:

- SEMS (Section *SEMS – the SIP and RTP processing Daemon*)
- redis (Section *REDIS – the Real-time Database*)

8.5.1 Process Management using Systemd

Systemd main system process management daemon manages processes that are started on both nodes independently and are not part of the HA pair management.

The following SBC SBC processes are managed by Systemd:

- **redis_events** - all events generated by ABC SBC and are sent to the redis local in-memory database, see Sec. *Diagnostics Dashboard* for more details.
- **redis_cs** - the redis local in-memory database for storing call state.
- **sbc-eventbeat-1**, **sbc-eventbeat-2** - this daemon connects to the redis event database and transferres the events to remote ABC Monitor.
- **sbc-repl-pcaprec**, **sbc-repl-pcaprec2** - traffic logs and recordings files replication to remote ABC Monitor.
- **stunnel-rsync** - provide TLS tunnel for traffic logs and recordings files replication to remote ABC Monitor, if connection via TLS is enabled.
- **sbc-tcpdump** - this is a continuously running tcpdump process that listens on all configured and enabled SBC signalling interfaces and captures the SIP packets as PCAP files.
- **syslog-ng** - standard Syslog-NG daemon used for filtering and storing log messages.
- **mariadb** - SQL server instance used as a storage for SBC configuration
- **sshd** - standard OpenSSH daemon used for remote console login
- **nginx** - standard nginx web server, which works as front end for ABC SBC GUI, xmlrpc access, tryit webrtc client page (if enabled) and websocket redirect (if enabled).
- **monit** - checks for high system load or CPU or memory usage or low disk space and creates alert events and sends email notifications.

- **snmpd** - SNMP daemon providing interface for communication with remote SNMP monitoring systems. For instance, SBC measurements and counters can be queried by external monitoring tools.
- **sbc-pullconf** - service to check and pull new configuration from configuration master.
- **sbc-checknet** - service to check for network buffer overflows and add alert events if errors found.
- **sbc-checkcore** - service that checks for any coredumps and creates diagnostic data tarball and adds alert events if found.

Every service is monitored by systemd and automatically started in case of any failure. A particular daemon can be manually stopped and started by:

```
% systemctl stop <service name>
% systemctl start <service name>
```

8.5.2 SEMS – the SIP and RTP processing Daemon

SEMS is the most important daemon as it processes the signalling and media traffic. It loads the settings from the configuration files and the SBC rules from the Mariadb database.

- configuration files: *"/etc/sems"* directory
- log file: *"/var/log/frafos/sems.log"*

To check whether SEMS is correctly listening on all configured SBC interfaces, see the output of **netstat** as shown in Fig. *Checking SEMS with netstat*.

```
[root@sbct2 ~]# netstat -tlupan | grep sems
tcp        0      0 127.0.0.1:8090          0.0.0.0:*               LISTEN      14278/sems
tcp        0      0 127.0.0.1:54242         127.0.0.1:705           ESTABLISHED 14278/sems
udp        0      0 127.0.0.1:5040          0.0.0.0:*                           14278/sems
udp        0      0 192.168.1.154:5060      0.0.0.0:*                           14278/sems
udp        0      0 192.168.1.153:5060      0.0.0.0:*                           14278/sems
udp        0      0 192.168.178.142:5070    0.0.0.0:*                           14278/sems
```

Fig. 48: Checking SEMS with netstat

8.5.3 REDIS – the Real-time Database

Redis is used for data replication between active and standby machines. On an active machine, it is running as "Masters," and on the standby machine as "Slaves".

- configuration file: *"/etc/redis.conf"*
- log file: *"/var/log/frafos/redis.log"*

The administrator can check the status of redis and which rule the process is having (and role) with:

```
% redis-cli |  grep role
```

The content of the redis database (the description of its records is out of the scope of this document) can be displayed using:

```
% redis-cli keys "*"
```

This command can be useful for checking whether data replication is correctly working by comparing redis content on active and standby machine.

8.6 Additional Sources of Diagnostics Information

The following additional sources of management data may be also used:

- traffic monitoring and event tracking described in Section *Overview of Monitoring and Troubleshooting Techniques*,
- remote monitoring described in Section *Using SNMP for Measurements and Monitoring*,
- process management described in Section *Command-line SBC Process Management*,
- logging concealed with call log in file as described in Section *Monitoring and Logging*.

When trying to understand some unexpected network or SBC behaviour the following facilities can be also helpful:

- Audio can be recorded as described in Section *Audio Recording*.
- CDRs, as described in Section *Call Data Records (CDRs)*, include useful information.
- The ABC SBC can be configured to send notification by email if some serious error such as exhausted disk space occurs. Configure the recipient email address under "Config→Global Config→Monitoring→Email for sending alerts". Configure SMTP server to which the emails will be passed under "Config→Global Config→Monitoring→Mailserver for sending alerts" if you need to use external mail server, or use default "127.0.0.1" to send the alert emails directly from the ABC SBC server. Configure various thresholds for alerts based on high system load, memory used, CPU waiting percentage and disk usage percentage under "Config→Global Config→Monitoring".
- There is a hidden webpage that includes a snapshot of all important system information. It can be found in the web interface under the path /sysinfo/sysinfo.php . Always provide this webpage when communicating with customer support. The webpage can take several minutes to appear because it gathers extensive amount of system information.
- There is a hidden webpage that creates a compressed tarball file with a snapshot of all important system information, dump of ABC SBC database configuration tables and latest coredump file (if existing) and starts download of the file. It can be found in the web interface under the following path: /sysinfo/sysdata.php Please have this snapshots ready when approaching customer support. This diagnostic tarball is created also automatically when any new coredump file is detected. There is also another hidden webpage under the following path: /sysinfo/sysdata-latest.php This link initiates download of the latest already created diagnostic tarball file, without creating new one. The diagnostic files are placed in /data/sbc-sysdata directory by default.

Chapter 9

Securing SIP Networks using ABC SBC and ABC Monitor (optional)

9.1 SIP Security Principles: Collect, Analyze and Police

Like any other Internet-based service VoIP servers can be target of fraud attempts, denial of service attacks and abnormal operational conditions such as registration storms after recovery of a failed router with a user population behind it. These have become common with prevalence of the SIP technology for telephony and need to be dealt with on a daily-basis. A key function of the SBC is to fend off such situations so that the infrastructure behind the SBC and service for the end-users remains unaffected.

Administering any service securely always consists of three steps: **collect data**, **analyze it** and **police**. Each of these steps is a necessity and always requires human judgement of an administrator. This can be challenging with the sheer amount of data to be handled and may resemble looking for the proverbial needle in the haystack. Every day a public SIP service for one thousand subscribers generates as much as 7 GB in 13 millions SIP packets! Obviously making an administrator look at every single SIP packet is not feasible and the ABC SBC FRAFOS solution comes therefore with many administrative aids.

The first two steps, gathering data and analyzing it, can be purchased using various tools. FRAFOS however strongly recommends use of its ABC Monitor (see Section *ABC Monitor (Optional)*) because it has a unique access to internals of the ABC SBC and can report on many specifics not seen outside of it. The data gathered by the ABC Monitor known as **events** come from inside the ABC SBC and can therefore reveal information not visible to anyone else: plain-text signaling and media which is encrypted to the outside (always the case with WebRTC), internal information such as reasons why a specific SIP request has been dropped, or correlation of dialogs that are obfuscated to the outside using Topology Hiding.

The following real-world example shows a typical attack on a SIP service. The Figure *Screenshot of a monitored password-guessing attack* shows the course of the attack and defense against it. The attack started on March 22, 2016 at 1AM local time from an IP address located in Guangzhou, China. It consisted of attempts to register as users with numbers beginning with "122". The site was initially not taking any effort to fend the attack off, resulting in 1000 authentication attempts per hour. While the attacker didn't succeed in registering a URI protected using well-chosen passwords in this case, endurance or a weak password could have crowned his undertaking with success. Therefore at about 10:30 local time, the administrator took an action and locked out the attacker's IP address. It took exactly one hour until the perpetrator realized his tool was receiving no responses back and started sending from a different IP address. Now the SIP service administrator found out that a static policy is not good enough and enabled a dynamic policy that locks an IP address if too many failed authentication attempts come from it. The effect came instantly: the attacks were locked at transport layer and began to appear only shortly in hourly interval: that's the period after which the attacker changed his source IP address – few attempts were then observed in the ABC Monitor until the new source IP address was banned again.

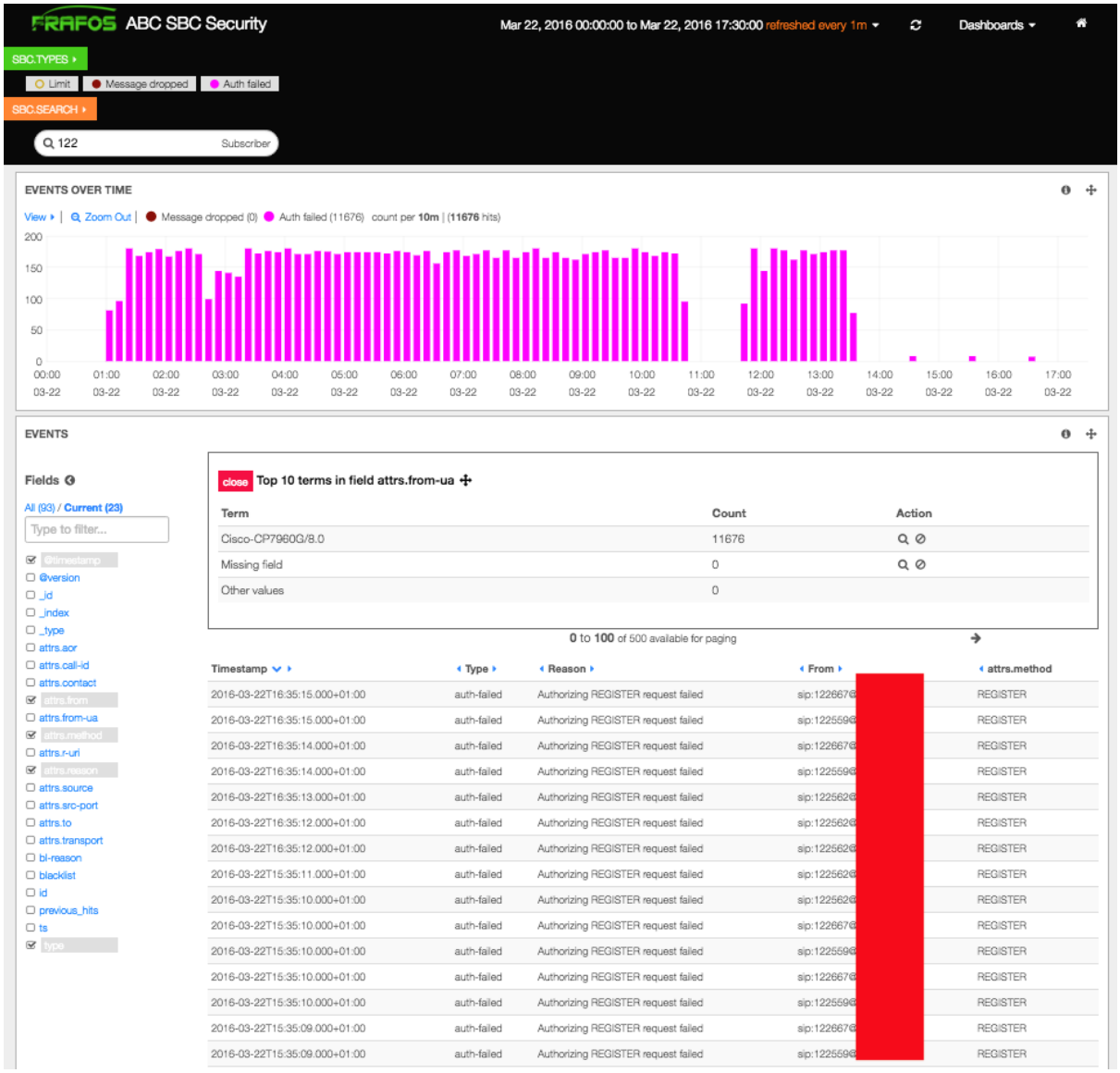


Fig. 1: Screenshot of a monitored password-guessing attack

This example is re-iterating the importance of the fundamental security principle: collect, analyze and police. If the site administrator didn't have good data about what's going on, he would be literally blind and the authentication attack could have remained unnoticed. All in all, two requests per second is not an excessive amount of traffic on a multi-thousand user-site, and the way the SIP protocol is designed almost every SIP request causes a 401/407 authentication challenge. Which leads to the second, analytical point. Usage data needs to be analyzed efficiently. The administrator needs to find out if there is anything going on at all, what are the specific patterns of an attack that can be used to fend it off, and who is the originator. The last step, fending the attack off, is the easiest once the nature of an attack is known.

These three facets of the security life-cycle are documented in the following sections. We will discuss them in the order a SIP packet encounters on its way. The first thing that happens to a freshly arrived SIP packet is it is processed by a ruleset that represent a site's security policy. We describe the available rules and practices for using them in the Section *Police: Devising Security Rules in the ABC SBC*.

Analyzing the security-related events using the optional ABC Monitor is discussed in the Chapter *Analyze: Finding Patterns in Events using the ABC Monitor*.

9.2 Police: Devising Security Rules in the ABC SBC

There is nothing more dangerous than security. Sir Francis Walsingham, Queen Elizabeth's Principal Secretary

The objective of the policing functionality is simple to state: Filter unwanted traffic as soon as possible before it causes harm. In order to achieve this objective, reasonable policies must be administered which permit legitimate and drop harmful traffic.

The delicate challenge is to differentiate between "friend and foe". Resolving this dilemma often requires a learning period – the administrator or an automated system on his behalf need to find out the presence of illegitimate traffic and its originator. An administrator can do this by analyzing traffic. The advantage of this approach is that human assessment of the situation can capture finesses a computer fails to see. This argument is for example the reason why air traffic control has never been fully automated – computers are still not trusted a judgement about abnormal situation.

The disadvantage of relying on humans is, not surprisingly, the human factor too. Humans may fail to see an abnormality in sheer amount of traffic and keep alert 24 hours a day. That's what computers are good at: they can look over gigabytes of traffic relentlessly, find patterns they have been taught to look after, and raise alarms any time of day as soon as they appear.

Therefore we at FRAFOS suggest that highest level of security of a SIP service is given when automated traffic filtering is combined with computer-aided human judgment.

In the following list we show typical attack types and also ABC SBC policies to deal with these.

- **Intrusion attacks** are attempts to obtain unauthorized access to a system or to a SIP user's account. They come by nature as an uninvited surprise at the most inconvenient time. The challenge is therefore to counter them as quickly as possible. In the Section *Automatic IP Address Blocking* we are showing how to **automate prohibition of malicious traffic** even before administrators do notice.
- **Harassing traffic** may be easier to detect and yet inconvenient to deal with. Unlike with real attacks, the harassing traffic is mostly an unintended side-effect of a broken implementation or configuration of some SIP devices. It doesn't try to masquerade or surprise yet if coming in large quantities, it may have the same devastating effect as a malicious attack. The capability to filter out such "noise" helps to reduce security risk, off-load the infrastructure, and focus on the traffic that matters. We show **how to block well-known sources of harassing traffic** at both IP and SIP layer in the section *Manual SIP Traffic Blocking*.
- **Unpriviledged traffic** is traffic that does not appear harmful yet it has not been explicitly authorized to use a SIP service. Such may not appear harmful on the first sight, yet it may be also an initial probing prelude to an actual intrusion attack. It appears therefore a wise idea to drop traffic which does not demonstrate appropriate credibility before it turns into a harm. This way **users exhibiting proper behaviour are prioritized** over users that don't. The simplest and yet most powerful credibility test is that of successfully completed SIP registrations. See Section *Blocking a User by his Registration Status* for guidelines how to use it. Also note that the credibility-test is extended to lower-layer by a generalized technique known as **grey-listing** (Section *Automatic Proactive Blocking: Greylisting*).
- **Excessive traffic** may have many root causes: Denial of Service (DoS), breach of service-level agreements, or SIP network misconfiguration. Regardless of the cause the results are always the same: quality of service (QoS) declines for legitimate uses. To prevent such QoS impairments, a site better chooses to set limits on SIP and or RTP traffic and drops traffic exceeding the limits. We show **how to shape traffic** in Section *Traffic Limiting and Shaping*. Traffic shaping is also important to discover some sort of attacks like SIP password guessing: if the attacking SIP device tries to masquerade as a legitimate user, the high signaling rate it needs for guessing will give it away.
- **Excessively long calls** are another irritating phenomena that needs to be dealt with in order to reduce a high-charge risk. Most often it is caused by SIP devices that do not terminate calls properly. Fraud attempts are also known that have been trying to gain maximum by running calls as long as possible. In Section *Call Duration Control* we explain **how to keep a SIP service robust against infinite calls**.
- **Improper content** in SIP signaling or SDP media can bring insufficiently robust SIP devices to failure. This situation doesn't happen so often because SIP devices typically do not have such processing capability like general purpose computers to be a real magnet for all kinds of viruses. Yet the situation changes as Android telephones come on the market and features offered by servers expand. Academics have already

described SQL injection attacks[1][2] : They crafted SIP messages which included SQL commands, and the SIP servers passed these to backend software. When the software is not sufficiently robust, opening a webpage to see a list of completed calls will also launch a potentially dangerous SQL query. If content of SIP and SDP is considered a risk, more aggressive mediation is needed. See Sections *SIP Mediation* and *SDP Mediation* for more information **how to filter SIP/SDP content**. Particularly header-field whitelisting may be instrumental for this purpose.

The ABC SBC offers several instruments for filtering undesired traffic. There are two types of: filters operating at IP/transport layer for the highest performance and filters operating at SIP layer when more sophisticated filtering criteria are needed. For example a well-known flooding attacker is best eliminated by filtering out all traffic from his IP address. On the other hand, if a single SIP user behind a SIP trunk IP misbehaves, blocking the whole trunk IP would be throwing the baby out with the bathwater. In such a case, the SIP-layer filtering would be a safer choice, albeit not that fast.

The IP-layer rules are managed from the administrative menu under **"System → Firewall"**. The screen offers a search box where one can look for an IP address to see if it is present on any of the lists, and then several firewall rule lists. The lists are ordered by precedence: the top lists are more manual, have higher precedence and override the bottom-placed lists. The top lists include Manual low-level rules, Exceptions to automatic blacklists, and Manual Firewall Blacklists and are described in Section *Manual IP-layer Blocking*. The bottom lists are generated in an automated way using built-in security assessment algorithms without administrator's intervention, can be overridden by the manual lists in the top, and are described in the Sections *Automatic IP Address Blocking* and *Automatic Proactive Blocking: Greylisting*.

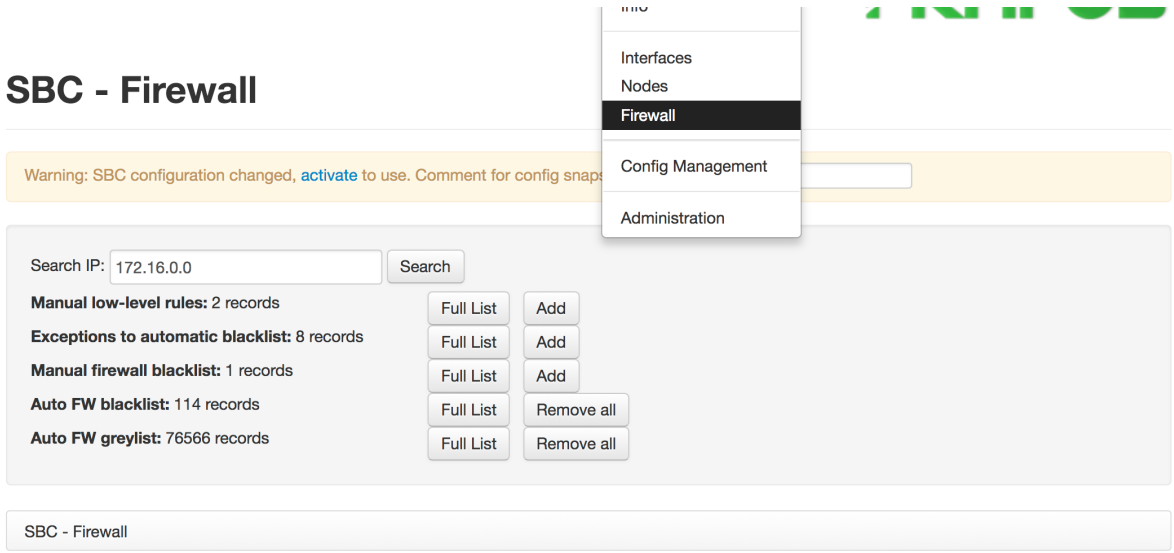


Fig. 2: Firewall Rules Management

SIP layer filtering is then described in Section *Manual SIP Traffic Blocking*. If binary yes/no policies seem too harsh, placing quota on the traffic may be a better answer, which is described in Section *Traffic Limiting and Shaping*.

[1] Geneiatakis, Dimitris, et al. "SIP message tampering: the SQL code injection attack." Proceedings of 13th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2005), Split, Croatia. 2005.

[2] Abdelnur, Humberto, and Olivier Festor. "Advanced fuzzing in the VoIP space." Journal in Computer Virology 6.1 (2010): 57-64.

9.2.1 Manual IP-layer Blocking

In some situations, e.g. if DOS attacks are encountered, incoming IP traffic may better be blocked already on the operating system firewall (iptables) level so that CPU processing power and memory is saved as the SBC processes don't need to handle the traffic.

The ABC SBC offers a graphical user interface to configure the firewall rules under **"System → Firewall"**. There are several rules list, the top-positioned rules list take precedence over the bottom rules list and are processed in the exactly same order as shown in the GUI.

If incoming packets do not match any of these rules, default rules apply. Traffic to signaling and media interfaces will be accepted if in the declared destination port range, traffic to administrative port numbers will be permitted on XMI and IMI interfaces, all other traffic will be dropped.

The top-most rules list is "Manual low-level rules" and it is a "swiss army knife" for firewall administrators. While it is the first-in-order list, we recommend to use it as the last resort due to extra complexity. Simpler rules such as "Exceptions" and "Manual blacklists" bellow are easier to manage and audit. Nevertheless the low-level rules my be still useful in situations when administrators wish to limit administrative access to well-known IP addresses or permit additional administrative protocols. These rules allow to specify IP flows using source and destination address and port numbers, and whether these flows should be accepted or dropped. That also means that attention must be paid to the order of these rules because it does affect the result. For example, the administrator can use the low-level rules block all traffic coming from the RFC1918 private IP address space as shown in Figure *Manual low-level Firewall Rules*. When a filtering criteria such as IP address or port number is left blank in the rule, any value in incoming IP packet matches.

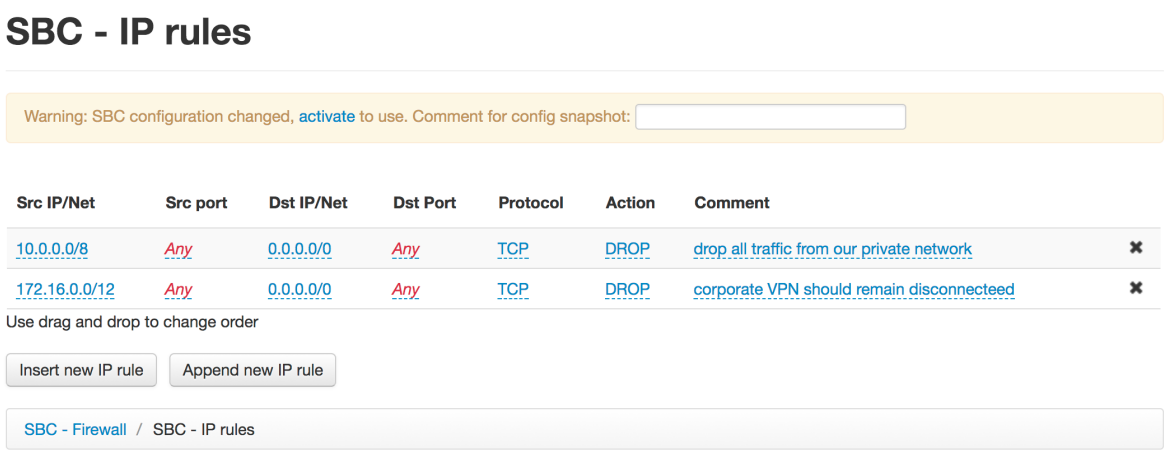


Fig. 3: Manual low-level Firewall Rules

The remaining firewall rules only refer to signaling (SIP and websocket) interfaces and are simple unordered lists of IP and subnet addresses.

"Exceptions to the automatic blacklist" are second in order and could also be called "Whitelists". They take precedences over any of the blacklists bellow. This is important to be able to override too zealous behaviour of automated blacklists. This is often the case when traffic of multiple users is coming from behind a single IP address due to NATs or a peering topology. Then the automatic blacklists triggered by a single user would block all others behind their shared IP address. Similarly a SIP site administrator may want to exempt himself from being auto-blacklisted, because his signaling tests may get him blacklisted. Consequently, he would not be even able to open an SSH session to the ABC SBC.

For example a single misbehaving URI would otherwise block an IP address and all other URIs behind it. In such a case, it makes sense to exempt this address from automated blacklisting and address the problematic URI traffic at SIP layer. Example of such a "Whitelist" is shown in Figure *Exceptions to the automatic blacklist*.

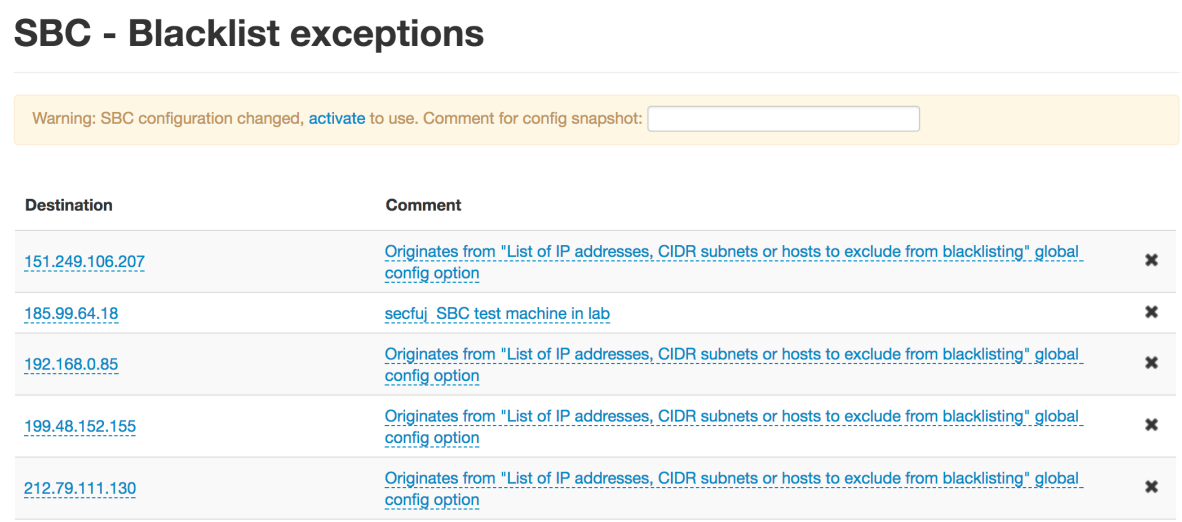


Fig. 4: Exceptions to the automatic blacklist

The third in order before automatic rules is "Manual Firewall Blacklist" that can disable traffic from an IP address or subnet even before it reaches any kind of SIP processing logic. This may make sense when a DoS attacker is detected whose traffic is better disabled as early as possible. Example of such is shown in Figure *Manual Firewall Blacklist*.

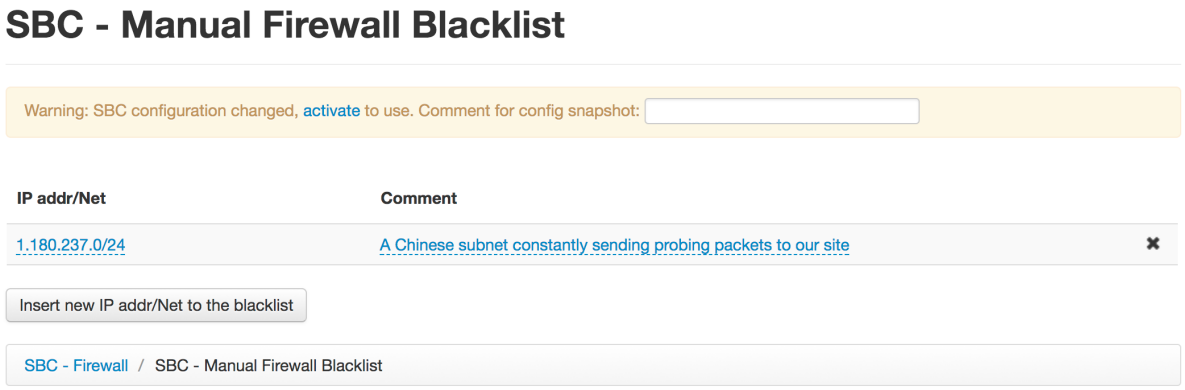


Fig. 5: Manual Firewall Blacklist

The next firewall lists, automatic blacklist and greylist, are populated in an automatic way by ABC SBC, and can only be flushed by administrator. The are described in the subsequent chapters *Automatic IP Address Blocking* and *Automatic Proactive Blocking: Greylisting*.

9.2.2 Automatic IP Address Blocking

The ABC SBC implements an automated protection process for SIP-layer close-to-real-time detection and IP-layer elimination of offending SIP traffic. This combination provides application-aware assessment with lower-layer performance and helps to eliminate offending traffic without manual administrator intervention. This level of automation cuts the detection-reaction time to almost real-time reactiveness.

A picture tells more than thousand words: The Figure *Number of Events with and without Automatic IP Address Blocking* shows the profound effect of automated blocking. The event timeline begins under protection of automated blocking in a calm way with about fifty events a minute. When at 23:30 the administrator turns off the automatic protection, the offending traffic finds its way and builds up rapidly. One hour later, 2500 events are

already reported every single minute, most of them failed authentication. This unfavorable status remains until the protection is re-enabled. Then, it takes less than five minutes until order is restored again.

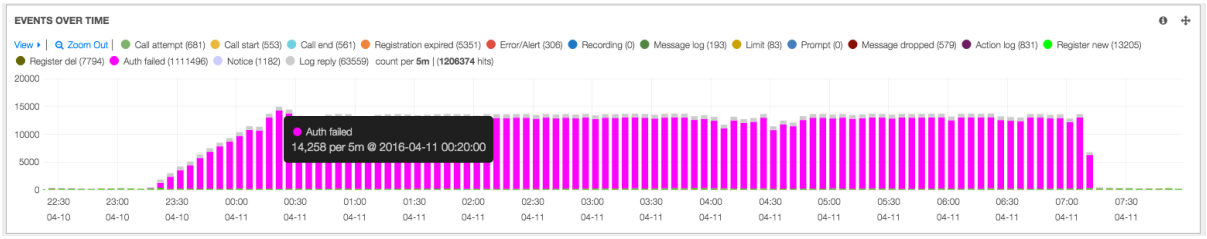


Fig. 6: Number of Events with and without Automatic IP Address Blocking

Intrusion attacks, by their very definition and purpose, come uninvited. Sometimes they may try to masquerade themselves in a way that the offending traffic looks innocent: Low-pace, using names of legitimate SIP device types. A human reaction may be too slow to identify such an attack. Therefore the automated process comes in: It acts before a human administrator could.

The ABC SBC protection process is based on the following empirical observations: Offending traffic comes in abnormal quantities, which are indicated by repetitive failures, these failures are linked to an IP address, and the IP address can be blocked. In other words, when some of the security-related events (see Section *Security Events*) come repeatedly from the same source, it is as good as certain we are dealing with an attack and need to isolate that.

Linking repetitive failures with an offending source is a quite reliable assumption. Singular failures do occur, for example if a softphone user types in a wrong SIP password an authentication failure event is reported. Yet if the same event is repeated many times, the more likely explanation is we have encountered a password-cracking attack. Leaving such an attack unattended creates a ticket for troubles. At the pace of 2800 authentication attempts per minute (45 per second) shown in our example, an attacker could crack a trivial password taken from Oxford Advanced Learner's Dictionary (185,000 entries) in less than 70 minutes!

Similarly, when a source continues to exceed traffic limits we are dealing with a Denial of Service attack, and when the ABC SBC is receiving repeatedly 403s from a downstream SIP service we know we are dealing with a scanning attack in which an attacker is trying to find a gap in a dial-out authorization policy.

In such a situation banning the originating source address at the OS layer is the safest way to keep the attack from the infrastructure. Care needs to be applied if in the network topology multiple users exist behind a single IP address: then legitimate users could be banned as well as the actual offender. This could be for example the case with peering traffic from behind a SIP proxy, or multiple users behind a single NAT.

The immediate effect of automated IP Address Blocking can be seen in Figure *Number of Events with and without Automatic IP Address Blocking*: At the very moment when it is enabled, the storm of authentication attempts calms down. It continues to appear briefly when either the attacker changes his IP address or the maximum "banning time" expires – then the detection mechanism strikes in again and the attacks vanish.

Once a source IP address is detected as a repeating offender, all of its traffic will be silently dropped. The list of all currently banned IP addresses can be found in the menu under **"System → Firewall → Auto FW blacklist → Full List "** together with the remaining time they are supposed to spend on the list.

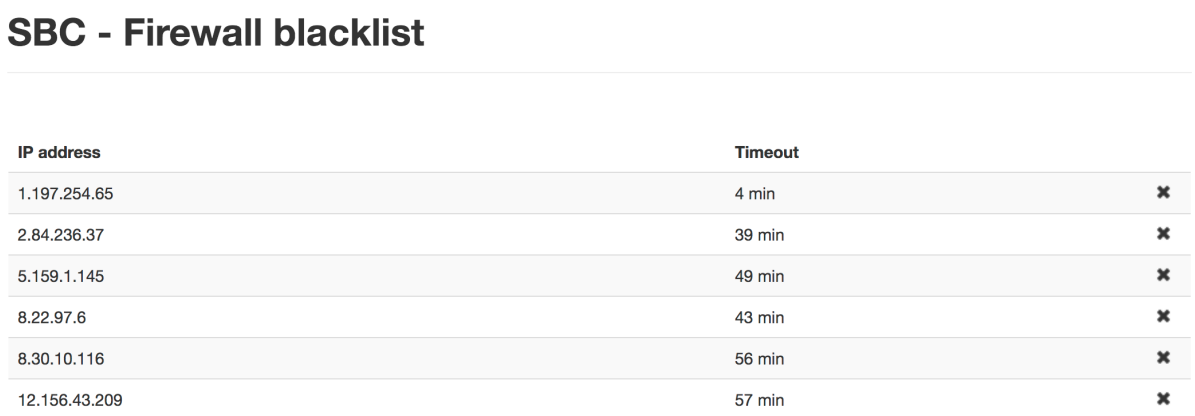


Fig. 7: Automated Firewall Blacklist

Scoring system

This effect is achieved by ABC SBC monitoring various occurrences that add to a "score" of a potential offender. To be banned, traffic of an offender must induce several serious events within a pre-configured period of time. Once the score is high enough to identify the originating IP address as "serial offender", the address is put on a blocking list and stays there for a pre-configured time.

The events that add to the score are all events documented in the Section *Security Events*:

- *limit* for excessive traffic,
- *message-dropped* for messages that the administrator chose to drop using the *drop* action,
- *auth-failed* for failed authentication attempts,
- *log-reply* for transactions which were declined by a downstream SIP entity,
- and significant errors to pass SIP compliance sanity checks.

SIP compliance sanity checks include:

- Request sequence number violation (based on CSeq checking).
- Request parsing errors:
  - malformed first line,
  - missing `Via`, `CSeq`, `From`, `To` or `Call-ID` header field,
  - Unparsable `Via`, `CSeq`, `From`, `To`, `Call-ID` or `RAck` (if included) header field.

Please note: sanity checks errors do not trigger any event.

Each of these events count as **1 offense**, with a **negative score of 1**.

The scoring system is implemented like a leaky bucket into which water is poured regularly. Once the bucket is empty, the offending IP is blacklisted:

- each new IP address starts with a bucket filled with a certain amount of water in it (**start score**).
- each offence decreases that score by 1.
- for every second passed, some water is poured into the bucket (**time bonus**).

If the start score is not considered, a certain IP is allowed `time bonus x time` offenses per `time`. For example, if the time bonus is set to `0.0001`, this means that `0.0001 x 3600 = 0.36` offense are allowed per hour. With `0.005`, this raises to `0.005 x 3600 = 18` offenses per hour, or `0.3` offenses per minutes.

The start score raises the score at the beginning so that the first offense does not cause blacklisting immediately (except if a huge time bonus is setup, which is not recommended), so that in normal cases it should be set to a value greater than 1.

Once an IP has been blacklisted, and the blacklisting expired, the score starts fresh as for a new IP.

If no offense has been registered in a certain amount of time (**time to remove entries**), the IP record is deleted, so that the next offense for that IP will reset the score to its starting value.

Given these settings, different strategies can be implemented:

- **trust strangers**: this strategy starts with a high start score (`> 5`), but won't allow any other offenses after that by using a very low or 0 time bonus.
- **forgiver**: the forgiver will forget about IPs that show a good conduct very fast (`< 300s`).
- **close watch**: the close watch will not allow much from the beginning (start score low; `~ 1`), allows an offence every now and then (time bonus `~ 0.0005 / s = 1.8 / hour`) and takes a long time to forget (time to remove entries `> 3600`).

Please note that these strategies can be combined together to allow for proper functionality without letting bad behaviour slip through.

Setting up automatic blacklisting

Automated blacklisting is turned off by default. To enable it perform the following steps:

- Turn it on. Under **"Config → Global Config → Firewall"** turn on **"Blacklist IP addr for repeated signaling failures"**. This will enable the automated blocking process that will process the "score" for IP addresses.
- Fine-tune it if necessary:
  - The options **"Signaling failures blacklist: IP address start score before any offense"** (recommended value: 2.8) and **"Signaling failures blacklist: rate per second used to calculate a time-related bonus between offenses"** (recommended value: 0.0005) in the same global configuration section allow to specify a threshold. When exceeded, the offending IP address will be blacklisted. The first parameter specifies an initial "allowance" that helps to overcome initial problems like forgotten password. The other parameter sets an error rate which can be tolerated over time.
  - The option **"Signaling failures blacklist: time in seconds to remove entries for which no event has occured from score calculation:"** states how long an IP address continues to be suspected after it produced its first security events. Recommended value is 7200.
  - The option **"Time in seconds to blacklist IP addr for signaling failures:"** determines how long an offending IP address stays on a blacklist. Recommended value is 3600.
- Define what occurrences add to the blacklisting score:
  - To include authentication failures and SIP protocol sanity checks, enable the options **Sanity** and **Auth** under **"Realms → Call Agents → Edit → Firewall Blacklisting"**. If this CA option is not set, the traffic coming from IP addresses within this CA will not be blacklisted.
  - Additionally scripting actions used for constraining undesired traffic may be set up to add to the blacklisting score. To enable the "drop" action to add to the score, check its "Blacklist by firewall if repeated" option as shown in the Figure *The Drop Rule Options*. To count originators of requests that were rejected by a downstream server, use the action **Log message for replies**, include message codes you are concerned about and turn on the option *Log to firewall blacklist*. The Figure *Scoring Rejected Requests* shows an example of such a rule intended to decline scanning attacks trying out calls to various telephone numbers. The guesses frequently fail and cause the replies with code 604. If this happens, the action *log message for replies* increases the blacklisting score.

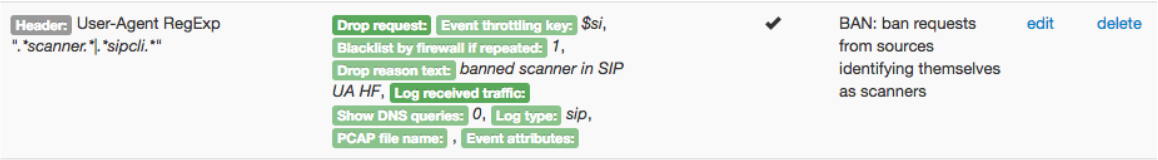


Fig. 8: The Drop Rule Options

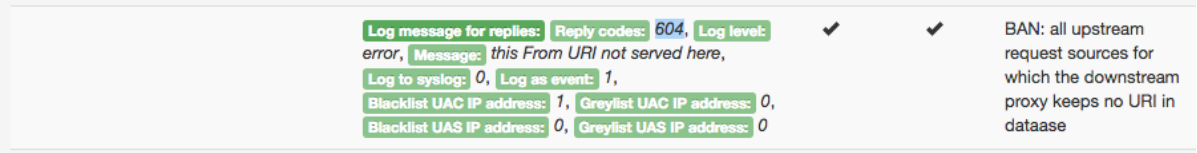


Fig. 9: Scoring Rejected Requests

Note again that blacklisting can impair legitimate users who share the same IP address with an offending user. This is often the case with NATs or a trunk Call Agent represented by a single IP address and a single user that is misbehaving. In such a case, it may make sense to turn off auto-blacklisting for such a Call Agent, and deal with the misbehaving URI using SIP-layer filtering as shown in Section *Manual SIP Traffic Blocking*.

9.2.3 Automatic Proactive Blocking: Greylisting

Sometimes an automated blacklisting policy may be too reactive in that it begins to block traffic sources only when they have been already "caught" misbehaving. An alternative automated and sterner policy, greylisting, may be used instead to block suspicious traffic coming from an interface preemptively.

The basic idea is very simple: Permit signaling traffic from unknown sources for only a temporary "probation period", accept it if some legitimate criteria is established within this period and block (greylist) it otherwise. In this case, all packets coming from the IP address will be blocked at OS layer for maximum performance. This concept is stronger than blacklisting in that it doesn't wait until a misbehaviour is spotted. An attacker trying to remain "under the radar" will not be tolerated any more. A single useless probing packet from his IP address to an ABC SBC signaling port will get him greylisted.

To enable grey-listing, you need to establish what makes legitimate traffic. An often used criteria is completion of authenticated SIP registration. To set up greylisting, proceed with the following steps:

- Turn greylisting on for an interface. Go to **System** → **Interfaces** → **Edit** → **Greylist**. At this moment signaling coming over this interface from an IP address will be dropped if the criteria does not establish its legitimacy within a strict time window.
- Define the legitimacy criteria. This is achieved using the actions **Log to grey list** and **Log message for replies**. The former immediately accepts a request source IP address. The latter does so later only when an answer with required status code comes back and can do so for UAC, UAS or both.
- Fine-tune greylisting global parameters if needed:
  - **time delay in seconds to give IP a chance to prove validity**,
  - **time period in seconds when IP can be blacklisted if repeats and did not prove validity**,
  - **time in seconds to keep IP on blacklist**,
  - **time in seconds to keep IP on whitelist**,
  - **additional ports or port ranges (a:b) to check in addition to signaling ports, space separated**.

Source IP addresses of cached registration bindings are implicitly accepted after receiving a successful response from the downstream registrar. This helps with a single administrative domain: an authenticated registration is quite a credible proof of sender's legitimacy.

However in scenarios with peering domains and other scenarios where SIP devices do not register, legitimacy of the senders must be established using some explicit criteria. To asses such a non-registering SIP sender, administrator must choose SIP transactions that demonstrate the sender is not an offender. This requires knowledge of a site's policy. For example, accepting an IP address based on an arbitrary 200-completed SIP transaction may be too relaxed, as any sender of a SIP OPTIONs "PING" packet that is "PONGed" would then qualify. Insisting on 200-completed INVITEs may be too harsh on the other hand, as a cancelled call attempt would result in graylisting the caller. Therefore the acceptance policy must be chosen with knowledge of what SIP transactions shall or shall not be accepted by the downstream SIP elements.

The qualifying SIP transactions are tagged using the "Log to grey list" and if dependent on the resulting transaction status the "Log Message for Replies" actions. When a transaction is processed using either action, and completes with a matching response code, then IP address of the SIP UAC, UAS or both will be accepted and will not be greylisted.

An example of such an A-rule is shown in Figure *Greylisting Rule Example: Accept 200 REGISTERs and selected non-REGISTER codes*. It accepts IP addresses from which REGISTERs come that complete with the 200 status code, and any other SIP requests that complete using some of the specified status codes. In all other cases, the IP address sending a packet to the ABC SBC will be blacklisted. That includes the cases when it is a non-SIP packet that doesn't even make it to rule processing, a REGISTER which doesn't result in a 200, and for example an INVITE which completes with the 604 code.

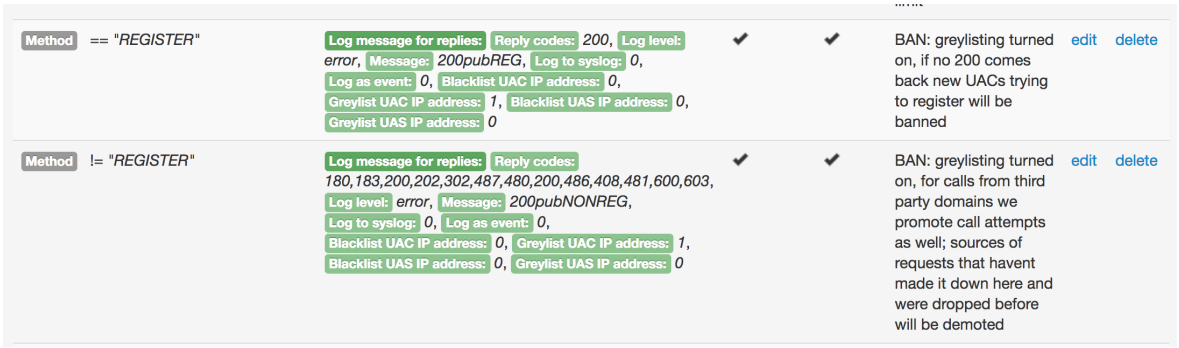


Fig. 10: Greylisting Rule Example: Accept 200 REGISTERs and selected non-REGISTER codes

If we wanted to craft a more relaxed policy which does not inspect SIP answers coming back, we could use the action **Log to Grey List** instead (Figure *Rule Example: Accepting an INVITE Sender's IP Address*). It accepts all IP addresses from which an INVITE comes. Its actual impact depends on where in the rules this action is placed. If it was in beginning of the rules, it would only block offenders sending non-SIP or non-INVITE packets to the signaling ports. Therefore it is typically placed after several rules that drop undesirable traffic, such as request from well-known scanners or unsolicited OPTIONs.

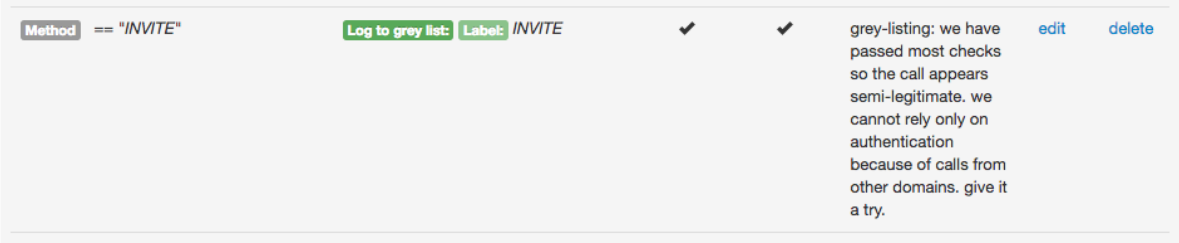


Fig. 11: Rule Example: Accepting an INVITE Sender's IP Address

We also have to care about outbound SIP requests. Answer packets coming back trigger the greylisting process and we need to have an acceptance policy as well. Typically it is quite simple under the assumptions that requests sent to outside express consensus to communicate with the outside IP address. Therefore installing a rule in C-rules to accept the destination address regardless of the response coming back will form a reasonable policy. Such a rule is shown in Figure *Rule Example: Permit UAS's IP Address for Any Replies*. A destination appears on the greylist only if it sends no answer.

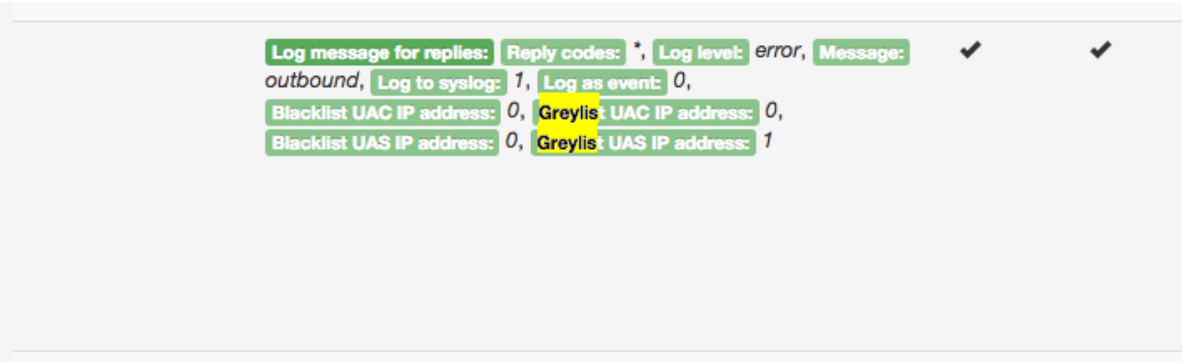


Fig. 12: Rule Example: Permit UAS's IP Address for Any Replies

Note that like with blacklisting, greylisting may have side-effects when there are multiple users behind a single IP address. A legitimate user who proves himself and promotes his IP address by the greylisting procedures makes traffic of other users behind the same IP also legitimate.

Blacklisting and greylisting may be used at the same time. In this case the side-effects of blacklisting will prevail as blacklisting goes first in the processing order. Then even if an IP address is accepted by the greylisting criteria, and a misbehaving user will cause the IP address to be blacklisted, all traffic from the IP address will be blocked.

It is also important to know that ABC SBC resets greylists upon every restart and starts re-learning them. This makes re-configuration and/or rapid failovers more robust against grey-listing innocent IP addresses. Otherwise a change of greylisting policies could fail to accept an IP address that has been already spotted under a previous policy. Similarly, a fail-over back and forth may also result in greylisting a legitimate IP address.

Checking the actual status of an IP address can be done on the administrative page **"System → Firewall → Search IP"**, from where one can also retrieve the full current blacklist and greylist.

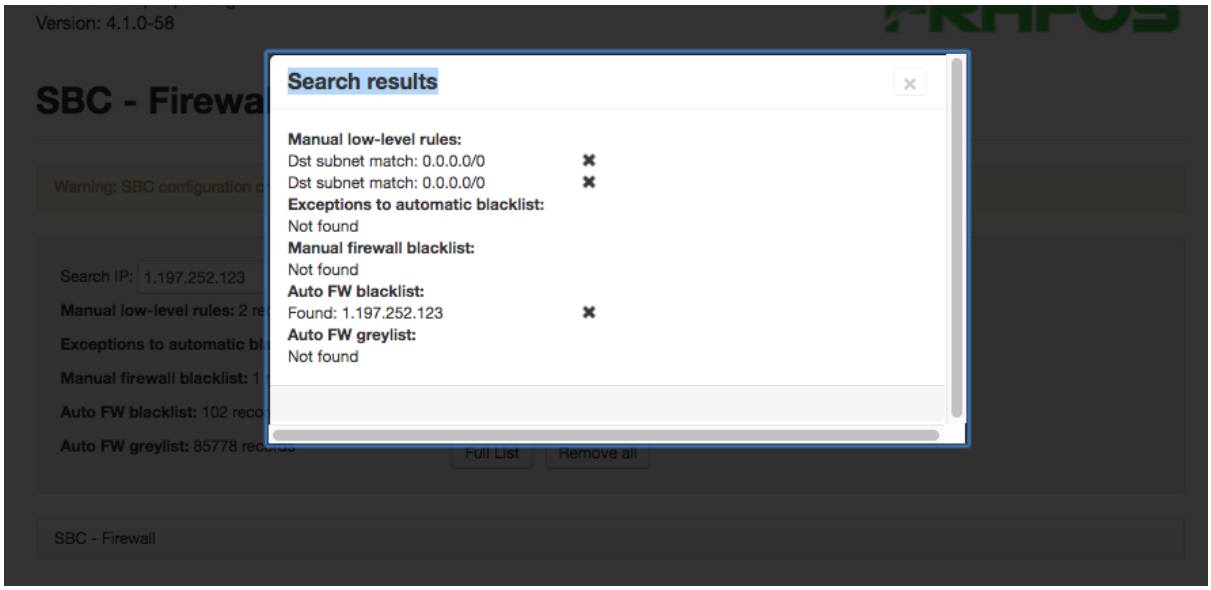


Fig. 13: Firewall Search IP Box Results

9.2.4 Manual SIP Traffic Blocking

The manual blocking is used to block well-known offending traffic using SIP-layer criteria. The SIP-layer blocking allows to establish SIP-layer filtering criteria, and it also allows to indicate to the upstream SIP client why a request is being denied using a SIP response code.

The reasons for using this type of blocking can be multifold: declining traffic from unsupported call agent types, refusing to process some unsupported applications like SIP for presence, or banning traffic from SIP users that have become unwelcome and cannot be dealt with using IP-layer blacklisting because they share IP address with other legitimate users.

A call can be refused silently or using a SIP response using either of the following methods:

- *Reply to request with reason and code*. This action declines a SIP request using response code and phrase. Optionally a header field may be attached to the response. Replacement expressions can be used in the response phrase and header field. Multiple header fields can be introduced by putting \r\n between them. An event of type "call-attempt" is generated for declined INVITEs.
- *Drop request*. This action drops a request silently and generates an event of type "message-dropped". Events can be grouped by a key in which case the events repeat within short interval of time (ten seconds) if their keys differ. If there is no key, the event does not repeat until offending messages stop to arrive for ten seconds.

If either action is executed, rule processing stops immediately and no further rules are processed. Neither do the requests count towards limits (see Section *Traffic Limiting and Shaping*) if the limits are placed behind the reply/drop actions.

The remaining question is how to discriminate between trusted and untrusted traffic. The ABC SBC can use any of its rule conditions described in Section *Condition Types*. The most often used conditions include:

- SIP header elements (Section *Blocking by User-Agent, From and Other SIP Headers Fields*)
- Source IP address (Section *Blocking by IP Address*)
- Registration status (Section *Blocking a User by his Registration Status*)
- Geographic origin (Section *Blocking by Geographic Origin*)

The following subsections documents the cases that are commonly used to filter out unwanted traffic based on different message elements. In the simple case, the tested elements are checked against fixed values like in the Figure *The Drop Rule Options* where the SIP requests are dropped if their Header Field User-Agent contains "scanner" or "sipcli". If the list of values to check against is longer, devising many rules may become cumbersome, use of provisioned tables is recommended as shown in the Section *Provisioned Table Example: URI Blacklist*.

Blocking by User-Agent, From and Other SIP Headers Fields

SIP request elements include many header fields upon which an administrator may make an accept/reject decision. For example, a SIP user may be found problematic and blocking his IP address is impossible because there are other legitimate SIP users behind the same IP address. In such a case it makes perfect sense to block all SIP requests with an offending address in the SIP From header field. Alternatively a whole domain can be blocked the same way. Conditions for this, From URI and From Domain, are available in ABC SBC rules, others are described in the Section *Condition Types*.

Not all header field names are available in the SBC rule conditions, and some may be even custom-made. Therefore there is also the possibility to refer to a header field by header name. That can be particularly useful when checking for some well known User Agent types that show their signature in the *User-Agent* SIP header field. Cases have been reported when this type of filtering has been used to block traffic from SIP devices with new untested firmware causing registration storms. Other common case is blocking well-known SIP scanners, one of such being known as "friendly-scanner". Their packets look like this:

```
OPTIONS sip:100@212.79.111.155 SIP/2.0.
Via: SIP/2.0/UDP 37.187.191.144:5064;branch=z9hG4bK-3414626242;rport.
Content-Length: 0.
```

(continues on next page)

(continued from previous page)

```
From: "sipvicious"<sip:100@1.1.1.1>;
 ↪tag=643434663666396231336334013337313833393332 35.
Accept: application/sdp.
User-Agent: friendly-scanner.
To: "sipvicious"<sip:100@1.1.1.1>.
Contact: sip:100@37.187.191.144:5064.
CSeq: 1 OPTIONS.
Call-ID: 383887304209490351968881.
Max-Forwards: 70.
```

A rule to detect, drop and record such requests from inbound (A) rules is shown in Fig. *Inbound rule for refusing calls from a certain user agent*.

Fig. 14: Inbound rule for refusing calls from a certain user agent

If the number of blocked elements become too long to have a separate rule for each of them, one can also utilize the provisioned tables as shown in the Section *Provisioned Table Example: URI Blacklist*.

Blocking by IP Address

It is possible to block a single IP address or multiple IP addresses matching a text pattern with actions configured with Source IP condition in the inbound (A) rule see Fig. *Inbound rule for refusing calls from a certain IP address*.

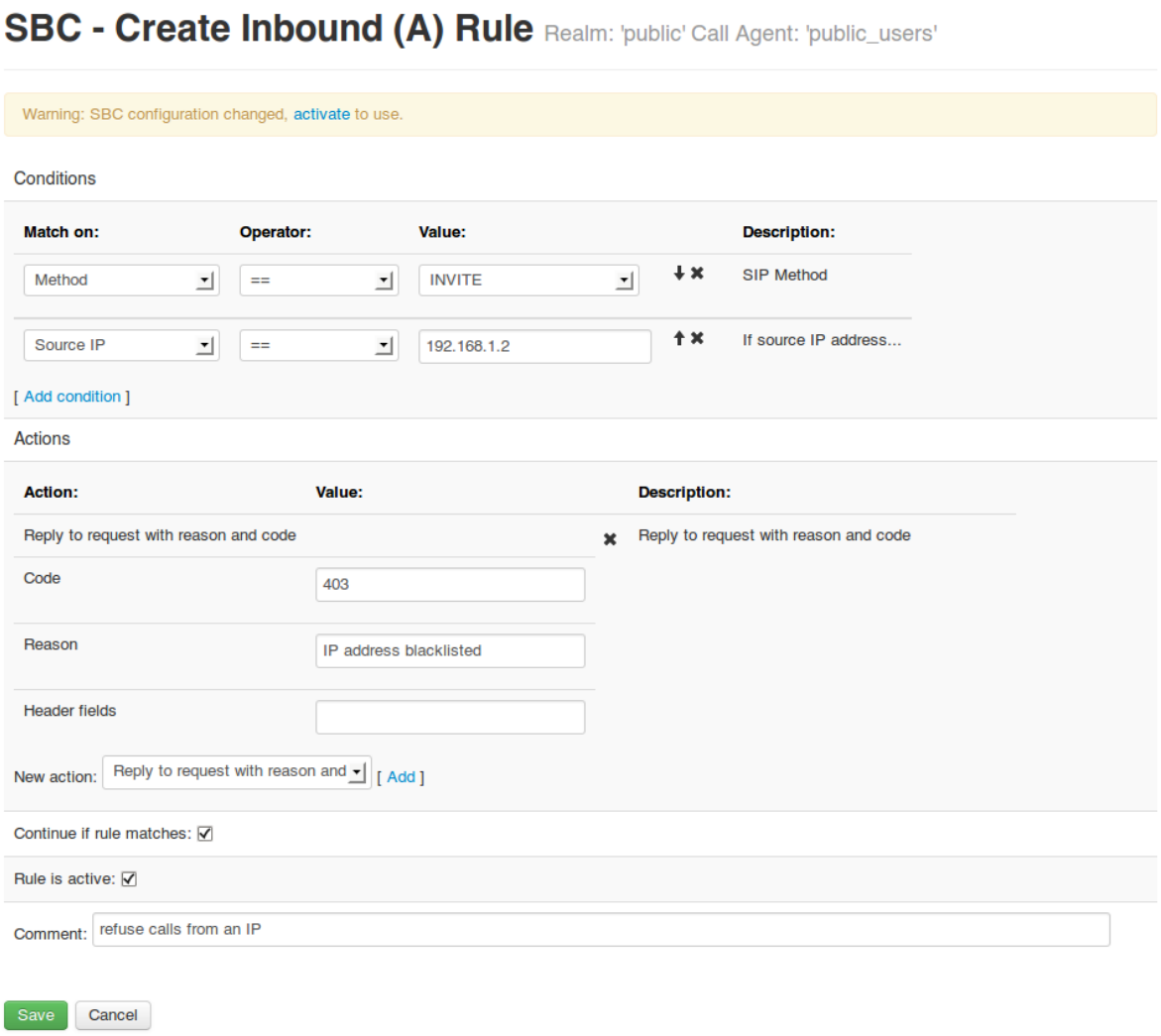


Fig. 15: Inbound rule for refusing calls from a certain IP address

Blocking by IP Address Range

The simplest way to block a range of IP addresses is to create a Call Agent for such an IP address range, see Fig. *Definition of a Banned Call Agent*, and create an inbound (A) rule for this call agent without conditions that will refuse all messages from it see Fig. *Rules for a Banned Call Agent*,

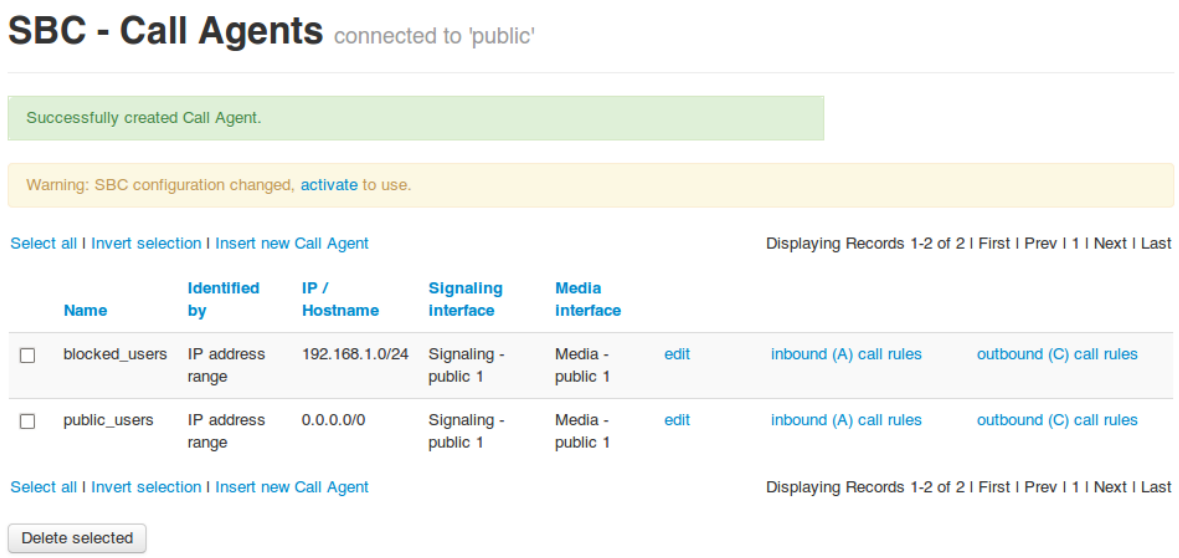


Fig. 16: Definition of a Banned Call Agent

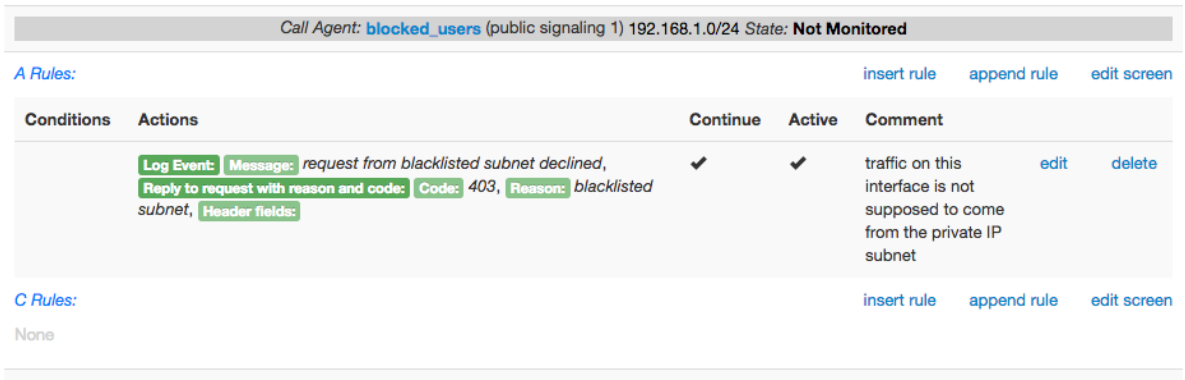


Fig. 17: Rules for a Banned Call Agent

Additionally this rule example uses the "Log Event" action to alert administrator of traffic violating his policy. (see Section *Diagnostics Events* for more details about using diagnostic events)

Blocking a User by his Registration Status

Inbound (A) rules offer a possibility to enforce an administrative policy by blocking the request (usually an INVITE) if its initiator is or is not registered by using condition *Register Cache*. It also can be used as some form or caller prioritisation if used together with CAPS limit. The test against the register cache is made using one of the following keys:

- From URI (AoR+Contact+IP/port)
- From URI (AoR+IP/port)
- Contact URI (Contact+IP/port)
- To URI (AoR)
- R-URI (Alias)

Such requests can be refused with **Refuse call with reason and code action**, see Fig. *Inbound rule for refusing calls based on registration status*.

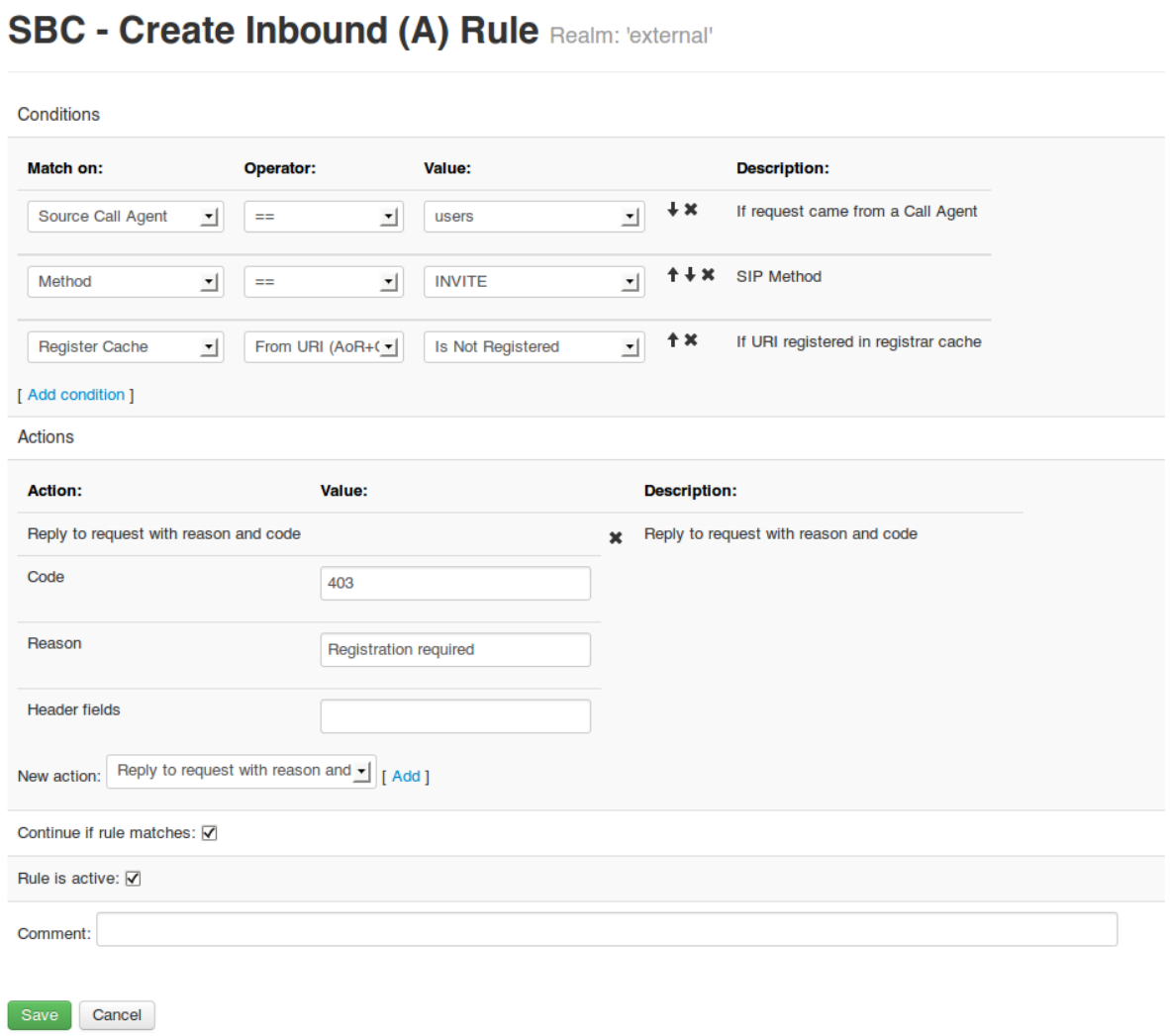


Fig. 18: Inbound rule for refusing calls based on registration status

Blocking by Geographic Origin

The ABC SBC can also block or otherwise discriminate incoming requests based on the country code of the region from which they are coming. The region is determined using a Geo-IP database from request's source IP address. The example here generates custom events when a request comes from France.

Fig. 19: Inbound rule for Reporting on French Request Originators

9.2.5 Traffic Limiting and Shaping

Like any other Internet-based service VoIP servers can be the target of denial of service attacks. By generating a flood of SIP requests a malicious attacker can overload the VoIP infrastructure. Such overload conditions can negatively impair established calls and calls in progress and need to be controlled. Similarly, authorized users of a SIP service can access the service in a way that reaches abusive dimension and needs to be controlled as well. For example, a provider offering a flat-rate service to consumers may find that whole PBXs are connected to the SIP accounts. This would result in losses since the pricing calculation anticipated different usage calculation. Therefore traffic control is also needed in such a situation.

The ABC SBC offers several forms of controlling SIP and RTP traffic which are described in this Section. These are implemented as rules which can be placed in A or C rule-basis. If used in inbound A-rules, the limitations refer to traffic **coming from** a Call Agent or Realm. If used in outbound C-rules, the limitations refer to traffic **sent to** a Call Agent or Realm. In either case the limitations only affect calls that passed the limitation action. Reversely, calls that have not been processed using a limit action are not subject to such a limit. To make sure that all calls within a realm or call-agent are subject to a limit, the action must be placed in beginning of the rules without any condition.

More often, the call limits need to be related to a subset of traffic. For example only one parallel call may be permitted per IP address. The criteria can vary depending on use-case and therefore the limiting actions have an optional variable key parameter. The key specifies which traffic portion the limit applies to, and can use replacement expressions (see Section *Using Replacements in Rules*). All messages (and no other) that have the same key count torwards the limit. Two often used keys are source address and combination of source address with From URI. The former (denoted as **"$si"**) checks all traffic coming from any single IP address against the respective limit. The combination of source address with AoR (denoted as **"sifu"**) allows that requests with distinct From URIs count against their own limits even from behind a single IP address – particularly useful when the IP address belongs to a PBX which serves numerous SIP addresses.

When the *"Is global key"* option is kept unchecked, the indexing key is scoped to the entity the rule belongs to (Realm or Call Agent). This means that the real key used to index the corresponding measurement is a compound of the indexing key and the entity. If, however, the key is declared to be global (by checking the *"Is global key"* option), the index is solely determined by the key entered in the *"Key attribute"* field. This means that if the same indexing key is used in another rule block (for example for another Realm or Call Agent), the limit will be applied jointly for calls on which this other rule block applies.

The traffic limiting actions also generate events when traffic does violate the limits – see Section *Security Events*. This is important for administrators to be able to notice such conditions and consider how to deal with such violations further: Whether to recognize these as illegitimate and continue blocking the originators, or to lift the limits if they find the above-limit usage has a legitimate reason. Only one event is produced for a detected excess of traffic limit, regardless of its duration. However, if the excess calms down and emerges again after ten seconds, a new event will be generated.

To make sure that an administrator can be alarmed even before a limit hits and starts to drop traffic, some of the traffic limit actions have the "soft-limit" option that creates diagnostic **notice** alarms but does not drop any traffic. Also, in the case that the traffic violates the "hard" limit repeatedly, the option "Report abuse" allows to block the offending traffic source – see Section *Automatic IP Address Blocking* for additional information.

The following call limit actions are available for use in A- and C-rules:

- **Limit parallel calls** - Set limit for number of parallel calls. New calls arriving in excess of this limit will be declined using the *403* SIP response. To make it easier to find the cause, the response includes a *Warning* header field with an additional hint: *Warning: Caps limit reached.* For example, to limit the number of parallel calls from the ABC SBC to a Realm or Call Agent, add a **Limit parallel calls** action to its outbound rules. Incomplete call attempt in progress whose context resides in memory also count temporarily towards the limit together with established call. That's an important security aspects: it makes sure new calls in progress are declined and cannot establish calls later that would exceed the limit. To limit the number of parallel calls from a Realm to the FRAFOS ABC SBC, add a **Limit parallel calls** action to the Realm's inbound rules. The action includes the following parameters:
  - *Limit parallel calls* – the actual number of parallel calls permitted.
  - *Key Attribute* and *Is Global Key* optionally define which partition of traffic counts towards the limit.

  - *SIP response code* and *SIP response reason* specify what type of reply is sent in response to a request that violated the limit. Optionally, header fields such as *Warning* may be added to the response using the *SIP Header* option. This option is intended to provide upstream client and troubleshooters with additional information explaining why a request is
  - *Soft-limit* value allows to specify the "soft" threshold which if exceeded will generate a diagnostic event.
  - *Report abuse* checbox makes occurence of a traffic limit violation count against automated IP address blocking score.
  - *SIP response code* and *SIP response reason* specify what type of reply is sent in response to a request that violated the limit. Optionally, header fields such as *Warning* may be added to the response using the *SIP Header* option. This option is intended to provide upstream client and troubleshooters with additional information explaining why a request is

- **Limit CAPS** - Set limit for SIP request rate. If the request rate exceeds this limit, new call attempts will be declined using a *403* SIP response. Note that when a request is authenticated using SIP digest, it results in two transactions, both of which count towards the CAPS limit. New dialog-initiating (e.g. SUBSCRIBE) and out-of-dialog (e.g. unsolicited NOTIFY) requests also count against the CAPS limit and will be dropped if they exceed it. SIP requests belonging to a dialog that has previously passed the limit test will all be accepted. Retransmissions do not count towards the SIP limit. The action includes the following parameters:
  - *Limit CAPS* – the number of permitted SIP requests per unit of time. These two values define the permitted signaling rate.
  - *Time Unit* – length of time unit in second. Even if the number of permitted requests grows proportionally with length of time unit and yields the same signaling rate limit, longer time units are more permissive as they can accommodate more intense bursts.
  - *Key Attribute* and *Is Global Key* optionally define which partition of traffic counts towards the limit.
  - *SIP response code* and *SIP response reason* specify what type of reply is sent in response to a request that violated the limit. Optionally, header fields such as *Warning* may be added to the response using the *SIP Header* option. This option is intended to provide upstream client and troubleshooters with additional information explaining why a request is being dropped.
  - *Soft-limit* value allows to specify the "soft" threshold which if exceeded will generate a diagnostic event.
  - *Report abuse* checkbox makes occurrence of a traffic limit violation count against automated IP address blocking score.

- **Limit Bandwidth (kbps)** - Set bandwidth admission limit for codecs. If current total sum of maximum bandwidth as signaled in SDP exceeds this limit, the signaling request will be rejected using a 403. For example, the limit of 30 kbps will reject any incoming INVITE that, among others, offers G.711 codec (64 kbps) in its SDP using a SIP 403 response. This type of limit only serves as initial admission control and does not guard the actual RTP usage. A sender is not hindered to send more RTP traffic than advertised in SDP unless the **Limit Bandwidth per Call** action is applied.

  The action includes the following parameters:
  - *Limit Bandwidth (kbps)* – maximum permitted bandwidth
  - *Key Attribute* and *Is Global Key* optionally define which partition of traffic counts towards the limit.
  - *SIP response code* and *SIP response reason* specify what type of reply is sent in response to a request that violated the limit. Optionally, header fields such as *Warning* may be added to the response using the *SIP Header* option. This option is intended to provide upstream client and troubleshooters with additional information explaining why a request is being dropped.
  - *Soft-limit* value allows to specify the "soft" threshold which if exceeded will generate a diagnostic event.
  - *Report abuse* checkbox makes occurrence of a traffic limit violation count against automated IP address blocking score.

- **Limit Bandwidth per Call (kbps)** - Set limit for RTP traffic per call. This action observes all RTP streams, video and audio, of a call, and if the actual traffic rate exceeds the limit, the RTP packets will be dropped. This action has the only parameter, the threshold value in kbps. RTCP traffic is not counted against the bandwidth limit and this bandwidth limit is only effective if RTP anchoring is enabled for the call. The limit includes RTP packets including RTP headers and excludes lower layer overhead (UDP,IP). For example for g.711 that makes 68.69 kbps (64kbps codec, 4.69 kbps RTP) and excludes 10.91 kbps overhead (3.13 RTP, 7.81 IP). For GSM the audio and RTP bandwidth is 17.69 (13 kbps GSM, 4.69 RTP), IP and UDP overhead is 10.94 kbps.

Note that limits are only applied to SIP requests that encounter the respective limit rule. That means that a newly introduced limit does not affect established calls. It also means that if call processing is stopped due to declining or dropping the call before the limit rule is evaluated, the declined call attempt doesn't towards the limit. Example of such rules where calls are declined before counted against a CAPS limit is shown in Figure *Order of Rules Matters: Dropped Calls Don't Count Towards Limits*.

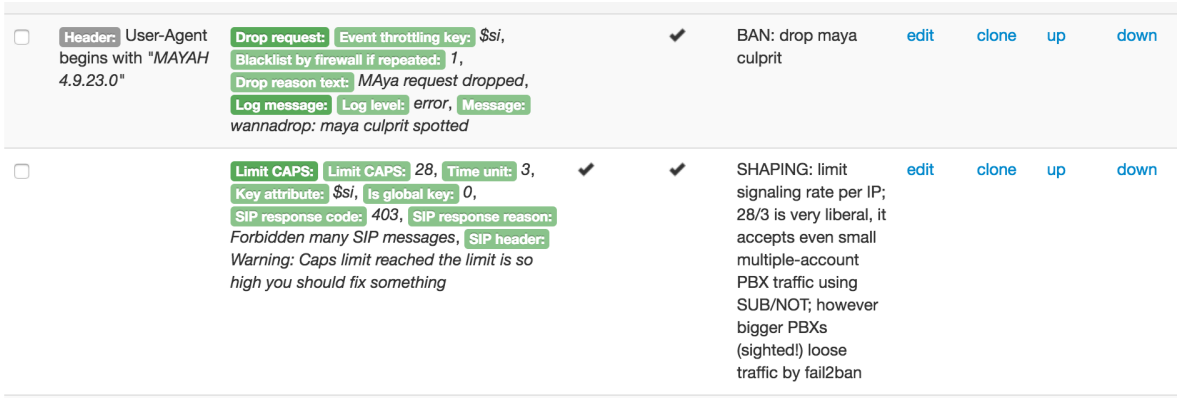


Fig. 20: Order of Rules Matters: Dropped Calls Don't Count Towards Limits

The most delicate part when setting the limits is finding the appropriate threshold values. Definition of appropriate values depends on what type of SIP User Agents are being used and how. Specific aspects causing higher traffic rates need to be considered to make sure that legitimate traffic will not be discarded:

- soft-clients often support SIP for presence (RFC3856). The amount of traffic, especially when such a client starts, can be high and grow with the length of the buddy list.
- Registration throttling (Section *Registrar off-load*) is often used to keep NAT bindings alive. The limit rate needs to be adjusted to the throttling rate.
- PBXs and Integrated Access Devices and most importantly trunking peers send traffic for many users from a single IP address.

Traffic Limiting and Shaping by Example

In the following example we implement a policy to shape incoming traffic for a public SIP service for personal use. The example is intended to be rather liberal and sets the threshold relatively high for the anticipated use to make sure it doesn't break some traffic-intensive use-cases.

We start policing VoIP calls in the first rule. To make sure that even a nervous caller attempting to reach a busy destination doesn't exceed his limits, we permit 10 requests every 30 seconds for every source IP address (the "$i" in the key parameter). Note that the actual number of call attempts may be lower by one half, since SIP authentication attempts preceding the actual call attempts count towards the limit as well and double the number of requests.

The next rule throttles registrations. We know that several popular consumer Integrated Access Devices (IADs) offer several SIP accounts. We want to make sure that the devices don't get locked out when they boot and send REGISTERs for all of their SIP accounts. We also need to account for the authenticating transactions. Therefore we set the limit to 10 requests every thirty seconds and key the limit by combination of IP address and From URI.

That means that the limit can only be exceeded by requests coming from the same IP address and bearing the same From URI. In other words, even if many REGISTERs come from behind a single IP address the limit will only be hit if they use the same URI. If the URI is registered from an IP address at a rate beyond the limit, parallel registrations of the same URI from behind a different IP address will not count towards the same limit.

Further we impose a general limit on all SIP transaction types. Especially soft-phones are known to send a lot of "noise": SIP PUBLISHes, SUBCRIBEs, NOTIFYs, OPTIONS and other request types. We permit 28 requests every three seconds from every single IP address.

Last but not least: we limit the number of parallel calls to 5 per IP address.

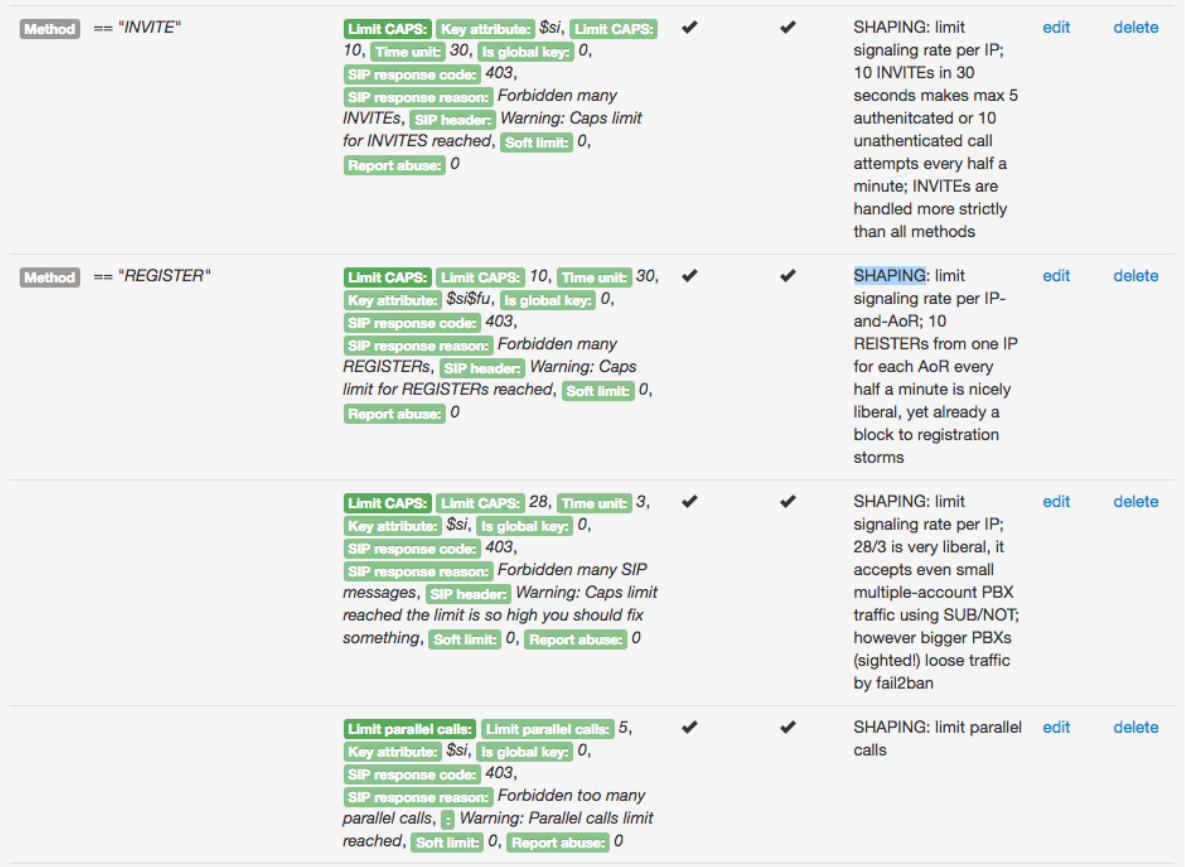


Fig. 21: Limit on number of Call Attempts per Second

Bandwidth limits by example

The ABC SBC can limit the bandwidth admitted for a calls' media streams. The action **Limit Bandwidth (kbps)** has as a parameter the bandwidth in kilobits per second to which the call should be limited, see Fig. *Example of Shaping Rules*. Attempts to set up calls exceeding this bandwidth will be rejected using a 403 response.

Fig. 22: Example of Shaping Rules

9.2.6 Call Duration Control

By limiting the maximum duration of calls one can on the one hand prevent "bill shocks" when some customer fails to terminate a call in a proper manner. Additionally, attackers might try to deplete the resources of the SBC by generating calls with long durations causing a saturation of the call processing capacity of the SBC.

Setting Call Length Limits

The **Set call timer** action sets the maximum duration of the call, in seconds. If a call exceeds this limit, the ABC SBC sends a BYE to both call participants and generates an event of the call-end type.

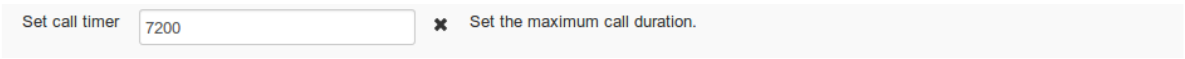


Fig. 23: Set call length

If this action is executed several times, the call duration will be the lowest call timer set, regardless of the order in which the actions are executed.

Controlling SIP Session Timers (SST)

SIP Session Timers (SST) is a mechanism defined in **RFC 4028** that can be used to make sure that calls are ended after a period of time even if one endpoint disappeared without properly terminating the call. This is especially important if calls are billed using data derived from the signalling messages, but also makes sure that unused resources are released properly. In order to achieve this, periodically a refresh of the SIP dialog is done by using a re-INVITE or an UPDATE. If the refresh request fails, e.g. if it times out, per the standard SIP mechanisms the dialog is torn down and related resources are released.

Note that the session refresh, i.e. the re-INVITE or UPDATE that is done here, is a normal re-INVITE or a normal UPDATE, so all the normal rules regarding the re-negotiating of the session apply. For example, a re-INVITE which is triggered by Session Timers may modify the session by selecting another codec or other codec parameters.

SIP Session Timers is a mechanism to negotiate which of the endpoints in the SIP dialog does this refresh. After the negotiation that happen with some specific headers (Session-Expires/x,Min-SE) in the INVITE or UPDATE and the responses to it, it is clear who has the refresher role.

The remote UA may leave it open who should be the refresher (by not including a refresher=uac or refresher=uas parameter in the Session-Expires/x header). In this case the ABC SBC has the possibility to select whether the ABC SBC should take on the refresher role or the remote UA should do it. This can be selected by the 'let remote refresh' option. It may be safer to do the refresh from the ABC SBC, as some UAs do not perform the refresh properly, even if they have said they would do it. On the other hand, if NATs are involved, there is no keepalive and the refresh interval chosen is too long, only the remote UA which is behind the NAT may be able to do the refresh thus reopening the NAT, in which case it may be safer to let the remote UA do it.

The ABC SBC supports SIP Session Timers even if one or both endpoints do not have support for Session Timers.

There are separate actions for enabling SST on the caller and the callee leg

- **Enable SIP Session Timers (SST) - caller leg**
- **Enable SIP Session Timers (SST) - callee leg**

The SST mechanism also negotiates the time after which the refresh is done. The timer parameters - proposed Session Expiration and Minimum Expiration, in seconds - can be set individually for each leg.

Setting RTP Inactivity Timer and Keepalive Timer

These two timers help to detect situations in which due to some network failure a phone call has already stopped. It requires media anchoring to be enabled.

It often occurs that a call party becomes suddenly unavailable and a call remains "hanging". This may happen for example due to a software error in a softclient or a disconnected IP network. To make sure such a call doesn't continue, an RTP inactivity timer may be configured. If configured and either party stops sending RTP, a call is discontinued by the ABC SBC after the preconfigured period of inactivity. Eventually an event of type "call-end" is reported with originator field set to "rtp-timer-terminated".

The timer can be configured under "Config → Global Config → SEMS → RTP timeout". If set to zero, the timer is deactivated.

To make sure that a peering SIP device using a similar kind of timer doesn't disconnect a call which just occurs to produce no media (voice inactivity detection, on-hold), the ABC SBC may also be configured to generate RTP keep-alive packets. If set to a non-zero value (in seconds), the ABC SBC sends keep-alive RTP packets periodically.

This timer can be configured under "Config → Global Config → SEMS → RTP keep-alive frequency".

Both timers can be also set on a call-by-call basis under parameters of media anchoring (see Section *Media Anchoring (RTP Relay)*).

9.3 Collect Events: Gathering Usage Data in the ABC Monitor

Knowledge is never too dear. Sir Francis Walsingham, Queen Elizabeth's Principal Secretary

An administrator can only craft reasonable security policies if he knows what is actually going on. He must have access to detailed history of SIP user behaviour, security incidents and unusual activities. This is indeed the purpose of "events" as introduced in Section *Events (optional)*. Events are detailed reports on user activity that encompass registration, call attempts, and security incidents.

Many individual events can identify need for administrator's attention. For example if a packet is dropped because it is coming from a SIP scanning software, an administrator may want to act and ban the source IP address.

Some events in isolation may alone not indicate a threat and need to be monitored by their quantity and trend. For example, an isolated authentication failure can be caused by a password typo during SIP authentication process. However if many such occur in series, chances stand high it is some kind of password-guessing attack as described in Section *Password Guessing Attacks*. Being able to recognize such repetition allows the ABC SBC to act automatically and even ban offending traffic without the human administrator's intervention. (see Section *Automatic IP Address Blocking*).

Most of the events are always produced by the ABC SBC, and administrators don't need any extra action to enable them. They just need to be able to understand and analyze them as shown in Section *Analyze: Finding Patterns in Events using the ABC Monitor*.

The rest of this Section is concerned with the cases when reporting events is optional and needs to be turned on explicitly to alert on possible departures from a SIP site's policy.

9.3.1 Reporting Security Events

As security events failures are reported when a particular administrator-defined policy is being enforced. The only exception is an authentication failure which is always considered a security threat.

The events are reported *only if* corresponding actions are executed and proper parameters are set. Therefore it is the rule conditions that primarily determine when to trigger an event.

The following table summarizes how rules must be set up in order to generate proper events. All shaping actions report limit violations if event reporting is enabled. So does the *drop* action when executed on an incoming SIP request, and *log-reply* when a request is rejected downstream.

Particularly the *log-reply* action is important as in some cases the downstream SIP elements may know better than the ABC SBC that a request is illegitimate. This is for example the case in a scanning attack when an attacker attempts to probe all possible SIP addresses. The ABC SBC is unaware of individual users and is not in position to repel such an attack straight off. However the downstream server knows subscriber details and can reveal by proper SIP response codes that the requests are for invalid destinations. It may respond back with 604 for non-existing users or 403 for forbidden addresses. This way the ABC SBC can infer from the response codes received from downstream that the upstream request originator should be better blocked.

Event Type	Required Action	Required Parameter	Additional Information
limit	Limit parallel calls	Report Abuse	*Traffic Limiting and Shaping*
limit	Limit CAPS	Report Abuse	*Traffic Limiting and Shaping*
limit	Limit Bandwidth	Report Abuse	*Traffic Limiting and Shaping*
limit	Limit Bandwidth per Call	none	*Traffic Limiting and Shaping*
message-dropped	drop	Blacklist by firewall if repeated	*Manual SIP Traffic Blocking*
log- reply	Log message for replies	Log to Firewall blacklist	

9.3.2 Setting up Diagnostic Events

Diagnostic events are also of great importance to the process of continuous refinement of security policies and bridging the gap between liberal and strict policies. A too liberal policy may lead to exposure of a security gap. On the other hand a too strict policy that filters all unknown SIP elements is likely to break some SIP features. Diagnostic events allow to strike a compromise, in which a policy remains open and diagnostic events report on suspicious traffic patterns. An administrator can then inspect these in details and choose whether they are legitimate and can be preserved, or whether they shall be better banned.

An example of such policy is reporting on call from unregistered users (see Figure *Rule Example: Report Calls of Unregistered Users*). If an administrator feels uncertain whether such calls are legitimate or not, he may initially just observe them. To do so, he places *log-action* in an appropriate condition and then watches the reported events. These include detailed information about the calls in question and provide the administrator with insights needed for further refinement of his policies. He may for example find out that the call attempts are coming from a peering domain and are perfectly legitimate. Or he may find that they have no traceable originator and should be better blocked.

The following table lists actions that can be used to provide customized reports on observed activities. The shaping actions can include an additional lower limit to report on high traffic before the "hard limit" is hit and traffic begins to be declined. The *action-log* can report on any conditions identified in the ABC rules: unexpected URIs, traffic from unregistered users, and anything else that can be captured by conditions specified in Section *Condition Types*. The *message-log* event is used analogously, in addition to the event details it also collects the actual traffic that triggered the event.

Event Type	Required Action	Required Parameter	Additional Information
limit	Limit parallel calls	Soft Limit	*Traffic Limiting and Shaping*
limit	Limit CAPS	Soft Limit	*Traffic Limiting and Shaping*
limit	Limit Bandwidth	Soft Limit	*Traffic Limiting and Shaping*
limit	Limit Bandwidth per Call	none	*Traffic Limiting and Shaping*
action- log	Log Event	none	*Diagnostics Events*
log- reply	**Log message for** replies	Log as Event	
message- log	**Log received** traffic	none	*Diagnostics Dashboard*

9.4 Analyze: Finding Patterns in Events using the ABC Monitor

See and keep silent. Sir Francis Walsingham, Queen Elizabeth's Principal Secretary

This section shows how the Monitor can be used when looking for security threats to a SIP service. Detecting presence of attacks and understanding their nature is a pre-requisite for devising proper policies that eliminate harmful and permit legitimate traffic.

Regular monitoring of a SIP service is proven to be the best way to keep operation smooth. Many administrators practice the simple and powerful habit to check their monitoring equipment as the very first thing in their working day. This section shows what to look for in the Monitor. Once a suspicious pattern is identified, the procedure is simple: keep filtering all regular events out until the events causing the pattern remain. Inspect their details and devise a proper policy.

In the following paragraphs we will walk you through the typical "stops" an administrator shall visit during his routing service check.

A good starting point is the **"Overview dashboard"**. Here a DoS attack can be discovered quickly as the security-related events gain dominance rapidly. Even Distributed DoS attacks can be spotted there because increased aggregate number of security events will reveal their presence, regardless how well masqueraded they are. This is shown in two 10-minutes examples in Figures *Total Number of Events in a Usual Situation* and *Total Number of Events when a Peak in IP Address Occurs Greylisting*.

Under normal operation, as shown in the upper part, the event types are balanced. There is a quite high number of greylisting events which is typical for a public SIP service exposed to scans from the public Internet. However, the number of these events still remains in the same order of magnitude as the other events. It is followed by the three registration events, also a typical situation for a public VoIP service in which telephones are turned off and on.

What should catch attention though is a situation in which an event type begins to dominate. This is the case in the lower diagram where greylisting events appear in unusual high quantities. That's good time to visit the Security Dashboard and find out more specifics.

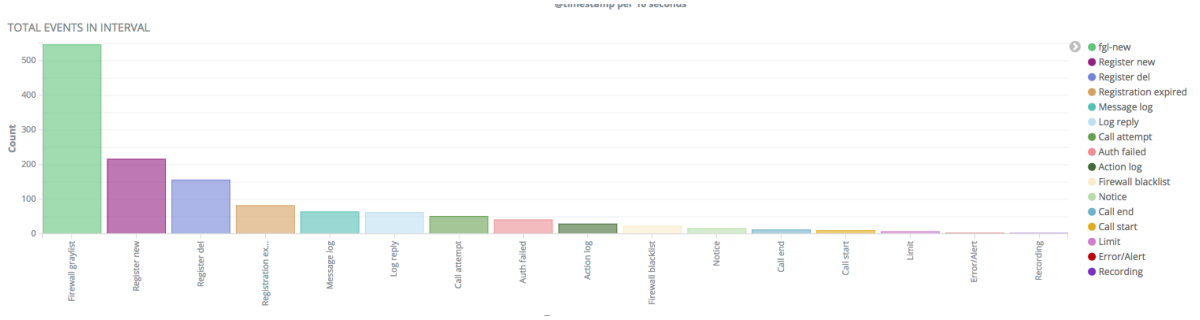


Fig. 24: Total Number of Events in a Usual Situation

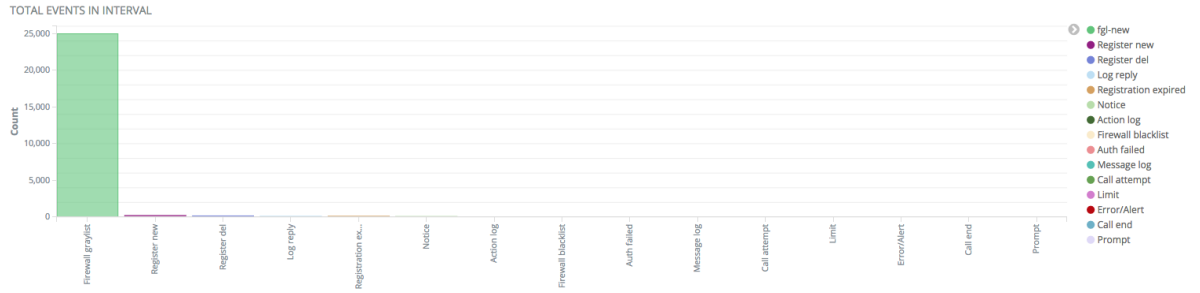


Fig. 25: Total Number of Events when a Peak in IP Address Occurs Greylisting

The **Security Dashboard** is introduced in more details in the Section *Security Dashboard*. The dashboard aggregates events that have relevance to security of a SIP service. Presence of authentication events points at password-guessing attacks (Section *Password Guessing Attacks*), presence of log-reply events at scanning attacks (Section *Scanning Attacks*), and presence of dropped-message events at rejected attempts to violate SIP site's security policies. So do Limit events unless they limit violation reaches an extend of a Denial of Service Attack (Section *Denial-of-Service Attacks*).

Not all SIP attacks have the ambition to ruin or gain control of a SIP service or a SIP user identity – some have the very simple motivation to find policy gaps that mistakenly permit phone calls. Such attempts are best spotted in the **Toplist Dashboard** as discussed in the Section *Dial-out Attempts*.

The following subsections detail on how to identify typical threats. Note that real incidents as recorded by FRAFOS are shown here that cannot be easily reproduced and may therefore include screenshots of previous ABC Monitor version with slightly different graphics.

9.4.1 Password Guessing Attacks

How would you feel if someone stole your password and was able to initiate and receive your phone calls, and all of it at your expense?

Password guessing attacks are really irritating: they are aiming at acquiring a user's password by guessing all thinkable variants of trivial passwords. The guesses are performed by machines. Once successful, the attacker can impersonate his victim and make phone calls on his behalf. The good thing is these attacks relatively easily manifest themselves by an abnormally high number of failed authentication attempts. If an attacker tries to stay under the radar, the total number of failures may not be apparent. Even then, however, the failures become easily visible when tied up to the attacker's IP address or geographic region. Such a situation is easily discovered in the Security Dashboard. A snapshot of the dashboard under attack is shown in the Figure *Events Reported during anEvents Reported during an Authentication Attack*. The toplist reveals that three of the ten most offending IP addresses come from the same subnet beginning with 200 and they are all clustered in South America.

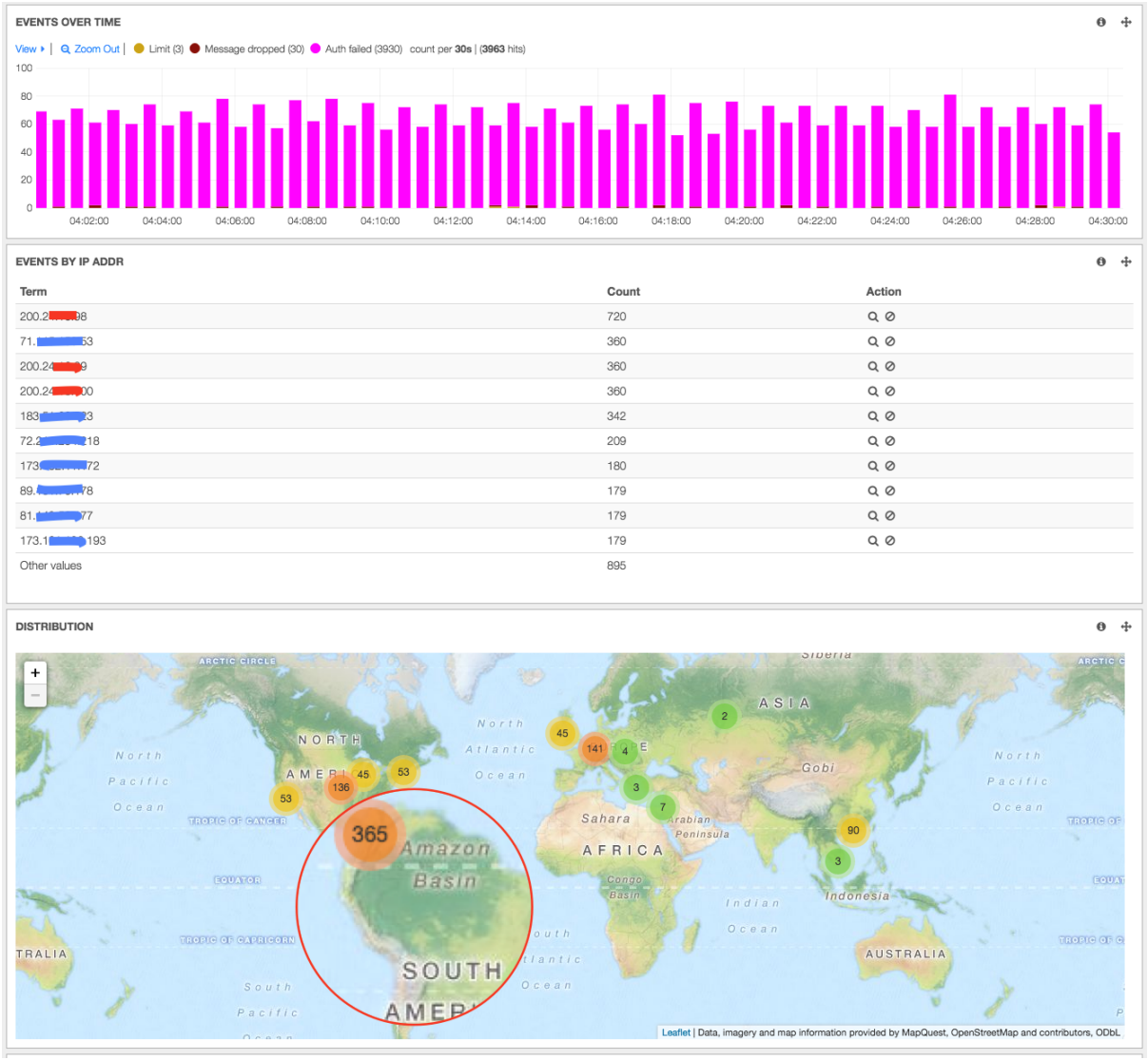


Fig. 26: Events Reported during anEvents Reported during an Authentication Attack

This intelligence is good enough to take counter-measures and lock the offenders by using some blocking techniques introduced in the Section *Police: Devising Security Rules in the ABC SBC*.

An investigative administrator may go further and study the attacker's background. He may look at event details, see if these are attempts to register or make a phone call, what URIs are being used, or even return to the Home Dashboard, filter events by IP address and find out about all other activities of the offender. He may also filter the events from this offender out to see if this massive attack hasn't overshadowed another less intense one.

9.4.2 Scanning Attacks

Would you be irritated if you found someone fiddling with your house door's lock? Then get some nerves with SIP services operating on the public Internet: This is happening every second and is called "scanning attacks".

Scanning attacks are attempts to send SIP messages to various SIP addresses to find out if the server is connecting calls to them. Very often, attackers dial-out a telephone number many times with various prefixes in attempt to find a gap in dialing plan that will let them through. The destination number is often a premium number that generates revenue for its owner. Scanning attempts typically result in an increased number of failed authentications when the SIP service policy requires authentication, or some 403s (Forbidden)/480s(off-line)/604s(no such URI) when the service is not serving a particular destination. Alone this information may be useful for an attacker – finding out that a destination exists may encourage him to mount a password-guessing attack against it.

Therefore it makes sense to check "Log Reply" events in the Security Console and find senders who are trying to reach many non-existing addresses or whose request often fail for other reasons. Such are often indeed originators of scanning attacks. We clearly see in the Monitor (Figure *Scanning Attacks Shown on Security Dashboard*) that the in the observed period the log-reply events peak up shortly.

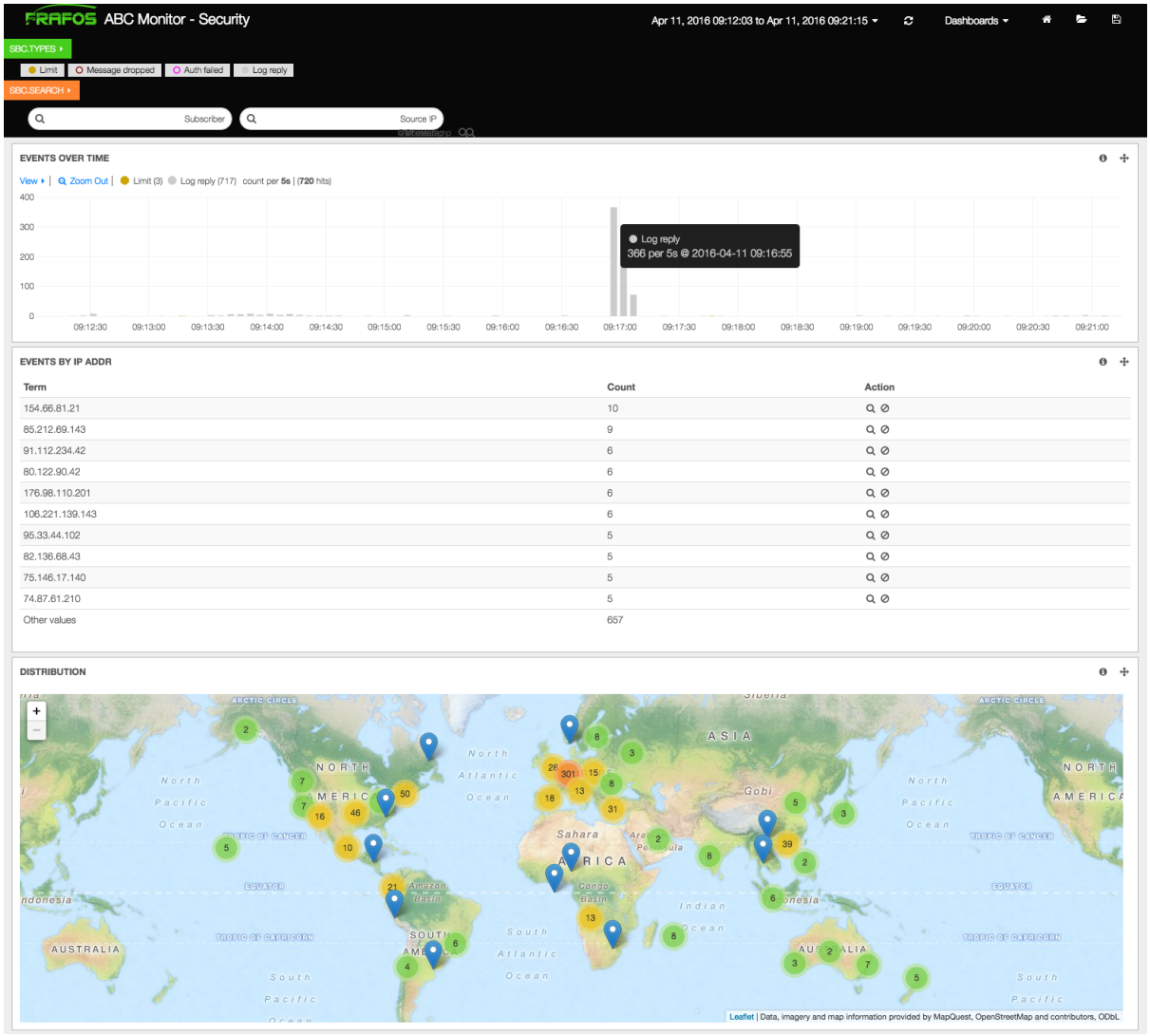


Fig. 27: Scanning Attacks Shown on Security Dashboard

The relatively flat distribution across IP address leaves us with a question whether that was a coincidence or an orchestrated distributed attack. In such cases administrator's insight is needed to judge the "Friend or Foe" dilemma. We discuss this further in the Section *Distributed Attacks*.

9.4.3 Denial-of-Service Attacks

Denial of service attacks (DoS) are simply excessive amounts of traffic targeted on a site with the sole purpose of impairing its service partially or completely. The ABC-SBC includes various counter-measures, probably the most effective one being the automated blocking described in Sections *Automatic IP Address Blocking*.

Presence of a DoS attack is best detected by the shaping actions that set traffic limits and report on their violations using *limit* events as documented in Section *Traffic Limiting and Shaping*. Adequate care is needed when devising the limits to match legitimate traffic patterns otherwise legitimate traffic could be shaped and reported on as abuse.

Once the limits are exceeded the offenders appear on the Security Dashboard. If the excessive traffic recurs and Auto-blocking is turned on, the offending IP addresses will be blocked.

An example of such a situation is shown in Figure *Excessive Traffic Captured on a Security Dashboard*. The *limit* event peaks come in quantities on timeline around 11:30 and in Hong Kong in the map. Further analysis, which is not shown in the Figure, reveals that a single User Agent Type is sending total of 5 requests per second using several distinct URIs. An administrator may then judge if his limits are set correctly and revealed an actual DoS attender, or if his limit criteria have been to strict.

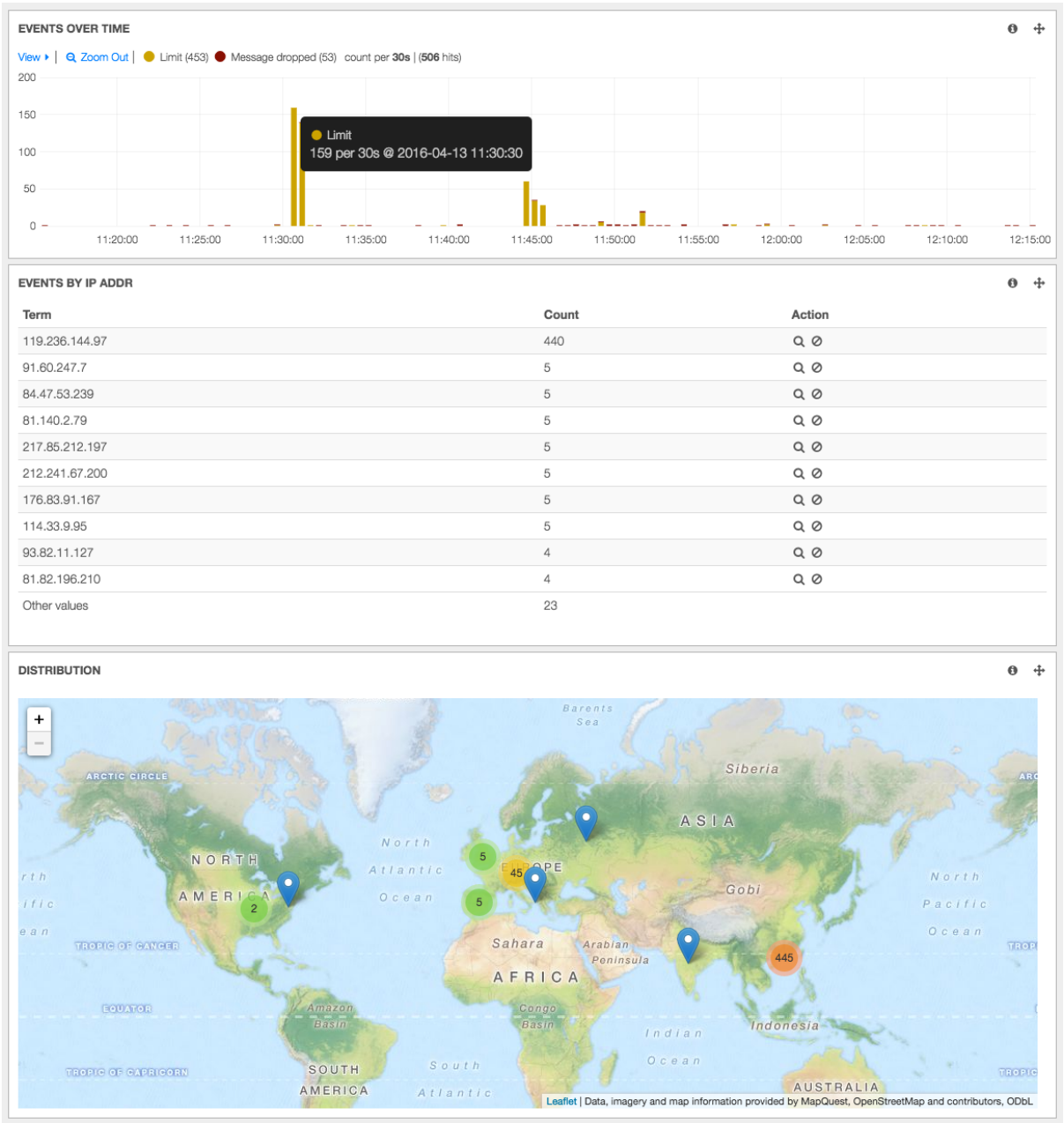


Fig. 28: Excessive Traffic Captured on a Security Dashboard

9.4.4 Distributed Attacks

Probably the closest non-computer analogy of a distributed attack is cross-fire when a target is exposed to an attack from many sides.

Distributed networking attacks are indeed a sophistication of other attack types that attempt to gather destructive force while remaining under radar screen. To obtain this explosive mix, the attacks are mounted from multiple sources. The most popularized distributed attack is the distributed denial of service attack (DDoS). However any other attack forms can be mounted in the distributed form and gain in invisibility and strike power.

The SBC can actually filter out substantial parts of a distributed attack on its own as shown in Section *Automatic Proactive Blocking: Greylisting*. The greylisting technique filters traffic which does not appear to do explicit harm yet it could have been probes prior to a real attack. It may be therefore worth to keep eye on the whitelisting and greylisting statistics. See the Figure *Network Statistics Dashboard Capturing a Failover Situation* in Section *Network and Statistics Dashboard*.

Still it may be useful to detect presence of a distributed attack. A way to examine validity of a situation is to compare the number of registrations (over 3 thousands in the example shown) and the number of greylisted IP addresses (around 70,000). The order of magnitude difference here shows there are many more IP addresses causing useless if not harmful traffic than those originating legitimate traffic.

Another place worth looking at is the Overview Dashboard and break-down of events by type, see Figure *Total Number of Events when a Peak in IP Address Occurs Greylisting*. While distributed attacks are more or less good in hiding their source, their presence can still be detected by their intensity, i.e. by number of certain event types that depart from values experienced under normal situation.

9.4.5 Dial-out Attempts

One of the very common attack forms is dial-out attempts. Technically it is the simplest possible attack and yet it can provide high gains to the attacker. It is simply trying to dial telephone numbers to see if the attacker can connect to them without authorization.

We have shown techniques that can discover such attacks in a generic way. If the attacker tries too hard and fast, his efforts will be exposed by the limit checks that serve the purpose of DoS detection (Section *Denial-of-Service Attacks*). If the attacker is not making good guesses about the service's numbering plan, his attempts will be discovered using the response-checking technique used also for discovery of scanning attacks (Section *Denial-of-Service Attacks*). The attacker may be smarter though: he can seek for unprotected telephone numbers at low pace and using some educated guesses about dialing plans.

So if the attacker manages to remain "under radar" we still have to find out that someone is trying. The simplest way is to visit the "Top Lists Dashboard" – it shows a number of attempts and completed calls sorted by URI. Abnormal number of call attempts or even completed calls will appear here, even if the signaling rate remained moderate and the caller "hit" valid phone numbers. Therefore the daily-check routine for an administrator shall include visiting the "toplist Dashboard" and inspecting events of the most active users.

The most active users appear in the "toplist dashboard" sorted in descending order. The example screenshot in Figure *Example: Toplist of Attempted and Completed Calls* gives us a quite clear picture: a user whose name begins with "fmat" attempts many more times calls than anyone else in the observed period of time. It is also unusual that the same user does not appear in the top-list of completed calls.

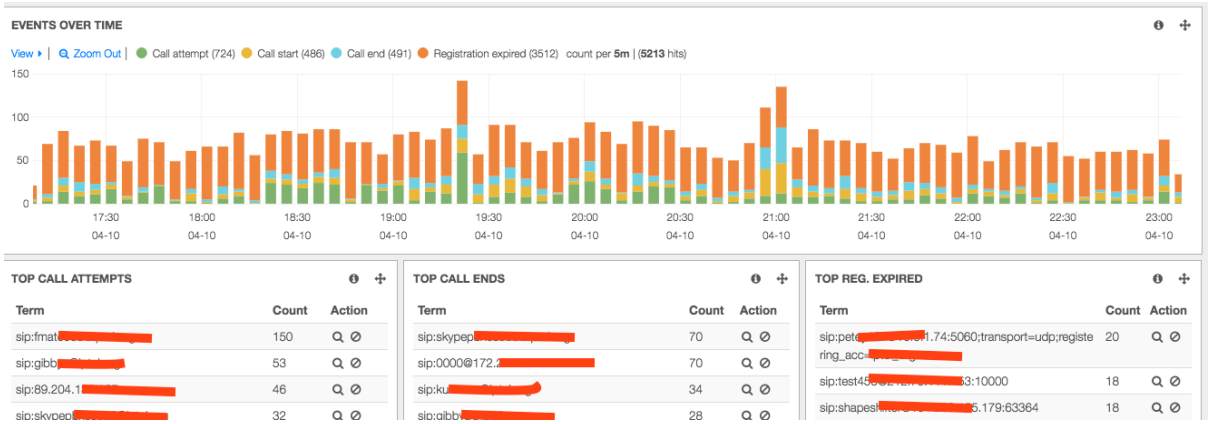


Fig. 29: Example: Toplist of Attempted and Completed Calls

To verify if this is a legitimate user, the same method can be used as introduced in Section *HOWTO Find a Needle in the Haystack: Iterative Event Filtering*: iterative filtering. When you narrow down the events to those originated by a suspicious From URI, the other toplists will show values relating to the specific originator. If for example, the destination toplists shows the same destinations only with varying prefixes, we know this was an attempt to break through. Example of such call attempt details are shown in Figure *Example: List of Attempted Destinations*. Obviously someone is trying to use a SIPCLI tool to call a number 48957372266 with varying prefixes (011, +, 9011, etc). The attempts are slow to remain under radar screen: they attempt every five minutes. They always yield the SIP response code 403 (Forbidden).

1–8 of 8

Time	type	attrs.from	attrs.to	attrs.source	attrs.from-ua	attrs.sip-code
November 11th 2017, 23:22:10.000	call-attempt	sip:55505@52.16.175.5	sip:56601148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 11th 2017, 23:18:08.000	call-attempt	sip:55505@52.16.175.5	sip:54401148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 11th 2017, 23:14:22.000	call-attempt	sip:55505@52.16.175.5	sip:53301148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 11th 2017, 23:10:58.000	call-attempt	sip:55505@52.16.175.5	sip:52201148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 10th 2017, 23:10:05.000	call-attempt	sip:55505@52.16.175.5	sip:801148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 10th 2017, 23:06:08.000	call-attempt	sip:55505@52.16.175.5	sip:901148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 10th 2017, 23:02:07.000	call-attempt	sip:55505@52.16.175.5	sip:+48957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 10th 2017, 22:58:10.000	call-attempt	sip:55505@52.16.175.5	sip:01148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403

Fig. 30: Example: List of Attempted Destinations

9.5 Practices for Devising Secure Rule-basis

While we have shown in previous sections how to police traffic, collect diagnostics information and analyze it there is still a remaining question: how to put all of this together in a consistent configuration using the ABC SBC rules. The way the rules are compiled can have significant impact on the logic of the service.

When devising the rule-base, the following important choices must be made:

- Whether to use **Media Control** or not. Relaying media (Section *Media Anchoring (RTP Relay)*) provides the ABC SBC with more control and insight into calls at the price of performance and media latency. Also it is a necessity when NAT traversal (*NAT Traversal*) needs to be implemented, IP addresses of infrastructural elements behind an ABC SBC need to be hidden, transcoding (Section *Transcoding*) or RTP-to-SRTP conversion (Section *RTP and SRTP Interworking*) is needed. If used, which is nowadays the default choice, the latency impact can be mitigated by geographic dispersion (see Section *Introducing Geographic Dispersion* for more information on what difference geographic distribution makes in a cloud environment).

- Whether to use **Topology Hiding** or not. Topology Hiding obfuscates signaling so that it is hard for an external party to find IP addresses of the infrastructural elements behind the ABC SBC . We are describing the rules to be used for topology hiding in the Section *Topology Hiding*. Note that obfuscation of SIP traffic may make its analysis quite difficult. If used tracing of traffic using ABC Monitor is recommended.
- How to organize policing rules. A reasonable practice is to start with rules that identify and instantly drop undesired signaling traffic before "heavier-processing" rules, such as media control or database queries begin. We show our recommendations in the Section *Devising a secure rule-base*.

9.5.1 Topology Hiding

Some service providers are worried about disclosing IP addresses of their infrastructure to third parties, attackers and competitors. Unfortunately the SIP protocol does such a disclosure in a grand style: SDP payload shows IP addresses from/which media is sent, Contact header-field shows the IP address of an end-point, and so does pre-RFC3261 CallID header-field. Via, Route and Record-Route header-field disclose the path of a SIP message exchange. Other standardized and / or proprietary header fields can also carry IP addresses.

Therefore service providers concerned about such disclosures prefer obfuscation of the respective SIP message elements. It needs to be pointed out though, that what makes life harder for attackers makes it similar hard or even harder for service operators. Correlating messages with each other for sake of troubleshooting is much harder when they are modified.

In the following subsections, we will review the default topology hiding behaviour and how to make it more transparent or more obfuscated.

Default Address Hiding

The default configuration of the ABC SBC tries to strike a good balance between the two extremes, full disclosure and full obfuscation. Already the back-to-back user-agent (B2BUA) design of the ABC SBC contributes significantly. The whole SIP path, as disclosed in *Via*, *Route*, and *Record-Route* header fields is split in two call-legs, each of them terminated by the ABC SBC. As result, each SIP dialog party sees the SBC as its peer in these Header fields. Additionally the ABC SBC by default rewrites dialog information (the triple CallID, From-tag, To-tag) so that IP address present in pre-RFC3261 implementations CallID is obfuscated.

If more signaling transparency is need than this default behaviour implements, transparent dialog ID can be enabled by an action as shown in the next Section. Also in some rare scenarios, the downstream elements in the SIP path may need to inspect the Via stack for the upstream leg. The ABC SBC reintroduces it when the following action is enabled:

Show Via

Transparent and Non-Transparent Dialog ID

The other concern is CallID – old-fashion SIP implementations pre-dating RFC3261 followed the RFC2543 specification and disclosed its address in CallID header-field. To make sure that the addresses do not get disclosed through this header-field when out-of-dialog or dialog-initiating requests are created by an elderly SIP User Agent, the ABC SBC can replace the CallID values with obfuscated values.

The choice whether to do is is administrator's. By default the ABC SBC does change the CallId Header Field value.

We recommend that administrators consider preserving the CallID for sake of troubleshooting. Leaving it unchanged makes correlation of incoming and outgoing SIP messages significantly easier. Enabling it is easy, what needs to be done is to place the following action in a Call Agent's or Realm's rules:

Enable transparent dialog IDs

Another advantage is that some non-standardized SIP extensions may want to take reference to a CallID value which becomes invalid once the ABC SBC changes it.

Hiding Addresses in Well-known SIP header-fields

When an operator is indeed concerned about disclosing internals of a Call Agent, the very step is to make sure that occurrences of the Call Agent's address in well-known header-fields are replaced with ABC SBC 's. Doing that is as simple as turning the Topology Hiding checkbox on under Call Agent's attributes. Once enabled all the following header fields will be rewritten accordingly:

- *P-Asserted-Identity* (**RFC 3325**)
- *P-Preferred-Identity* (**RFC 3325**)
- *Diversion* (**RFC 5806**)
- *History-Info* (**RFC 4244**)
- *Remote-Party-ID* (proprietary pre-3325)
- *Call-Info* (**RFC 3261**)
- *Warning* (**RFC 3261**)

Note that if the *Warning* header field is obfuscated, it is denoted using an additional *;topoh* parameter. This makes it clear that the address in the header-field is not genuine – otherwise a troubleshooter may be misled.

Hiding Contact Header in REGISTER

The *Contact* header is by default obfuscated by the SBC in all dialog-initiation transactions. Contact header field in REGISTER requests remains however untouched. If obfuscation is desirable, ABC SBC's register cache must be used that replaces the original Contacts with aliases.

The Section *Registration Caching and Handling* provides details about configuring registrar cache. This may be a reasonable option to be turned on alone for its "shock absorbing" and "NAT keep-alive" capabilities.

Hiding All Other Header Fields

Additional header-fields, standardized (Service Route **RFC 3608**) or proprietary, may appear and convey IP addresses. The ABC SBC only obfuscates the documented header-fields and doesn't interfere with others. If other header-fields are present and disclosure of IP address is a concern, the administrator can remove them at the risk of affecting the purpose they are serving. He can either remove the specific header-fields or use header field whitelist, i.e. remove all but well-known header-fields. SIP manipulation is described in detail in the section *SIP Mediation*, of particular interest is the action **set header whitelist**.

Concealing Media

Similarly like with SIP, the ABC SBC can put itself in the middle of the path and present itself to each call as its peer while hiding the other party. If the ABC SBC doesn't do it, IP address used for sending/receiving media will be seen in SDP and in the actual RTP packets.

To conceal the media sender/receiver, the following action must be enabled:

Enable RTP anchoring

The downside is that all media visits the ABC SBC, while increasing media latency and bandwidth imposed on the server. A detailed discussion can be found in the Section *Media Handling* .

9.5.2 Devising a secure rule-base

Developing a reasonable security policy may be a delicate task for a SIP service administrator. A too strict policy may too easily "throw the baby out with the bathwater" and impair legitimate traffic. The other extreme, a too liberal policy, may be too inviting for an attacker. Finding the right balance between serving users and protecting them from attackers requires understanding of the service goals and risks and drawing a balance between them.

The policy represented by the ABC rules also has performance implications associated with it. Some rules, such as database lookups, have higher latency and lower throughput than others.

We therefore suggest that policies are crafted in order of increasing complexity, starting with rules that instantly reply certain requests and continue to more complex rules. Basically, all undesirable SIP messages should be eliminated by rules in the initial rule-base part before processing for the accepted messages starts in the other part. The following subsections show examples of such rules in such order.

Shaping the Signaling Rate

It makes sense to begin processing with a check against SIP rate limits. Placing the check in the very beginning makes sure that all incoming SIP requests are checked against these limits including requests that are dropped by rules.

In Figure *Rule Example: CAPS Shaping* we are showing a simple rule example for sake of this Section. The rule checks all incoming SIP messages against a request rate and declines messages in excess of the limit.

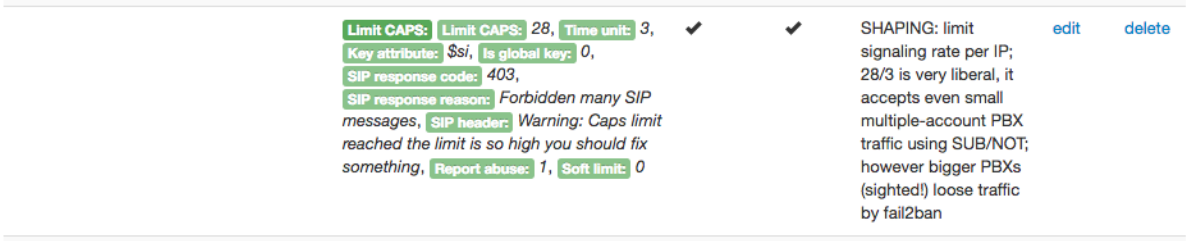


Fig. 31: Rule Example: CAPS Shaping

More sophisticated examples for shaping rules have been given in Figure *Limit on number of Call Attempts per Second* in Section *Traffic Limiting and Shaping by Example*.

Instant Responses

Many requests come that do not require any sophisticated decision making: the right action is just to send a reply instantly. The reply can be positive or negative. Positive replies are typically sent in answer to some SIP User Agents' SIP-layer NAT keep-alives. Negative answers are sent when requests request some unsupported service, do not comply to some local URI conventions, or come from a User Agent type known to malfunction.

The following rule examples show both cases: positive reply (Figure *Rule Example: Confirming SIP Keep-alives*) for keep-alive messages and negative replies to decline a request for unsupported Message Waiting Indication server (Figure *Rule Example: Declining an MWI Request*).

Fig. 32: Rule Example: Confirming SIP Keep-alives

Fig. 33: Rule Example: Declining an MWI Request

Dropping

With SIP requests who appear a security threat, dropping them silently is a safer choice than declining them. The less information an attacker gets, the harder it is for him to find a security gap in a SIP service. If an IP address is sending clearly offending traffic, it may even make sense to ban it entirely.

A typical reason for deploying such a restrictive rule is elimination of SIP scanner traffic. SIP scanners are automated tools that probe Internet address space to see if there are some SIP services running. Such tools are even publicly available[1]. When such a tool finds a responsive SIP service, it continues looking for legitimate SIP addresses and it may even proceed to mounting a password-guessing attack. Such attacks are real: Once you start a SIP service on the public Internet, it takes no longer than few hours until the first scanning packets arrive. Note however that filtering such traffic is only eliminating naively crafted attacks that advertise themselves as such. More sophisticated attacks will certainly not do it and must be detected and repelled using other methods such as traffic shaping.

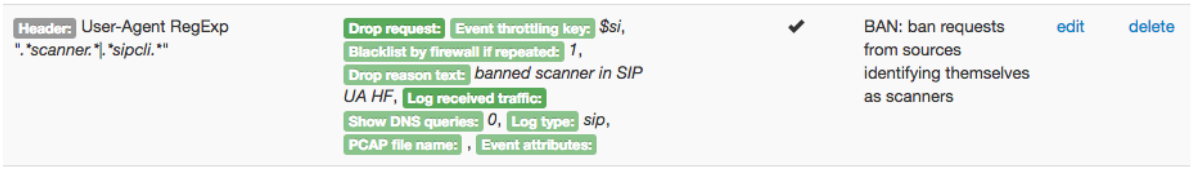


Fig. 34: Rule Example: Eliminating Traffic from SIP Scanners

The rule has an important option turned on: “Blacklist by firewall if repeated”. That means if the offending traffic appears repeatedly, the originator’s IP address will be blacklisted.

Database Checks

By now, we have eliminated most of the unwanted traffic: we have declined excessive traffic, gracefully handled SIP-layer keep-alives, declined politely messages for unavailable services and dropped obvious security threats. The remaining traffic has been reduced to a level where we can deploy more expansive policy checks and dig in database. Often there are offending users identified by their URIs. A straight-forward way to eliminate their traffic is to provision a list of such users and block SIP traffic if it comes from such users. Figure *Rule Example: Prohibited URIs* shows a rule that looks up SIP requests From URI in such a table and if the URI is found, drops the request silently.

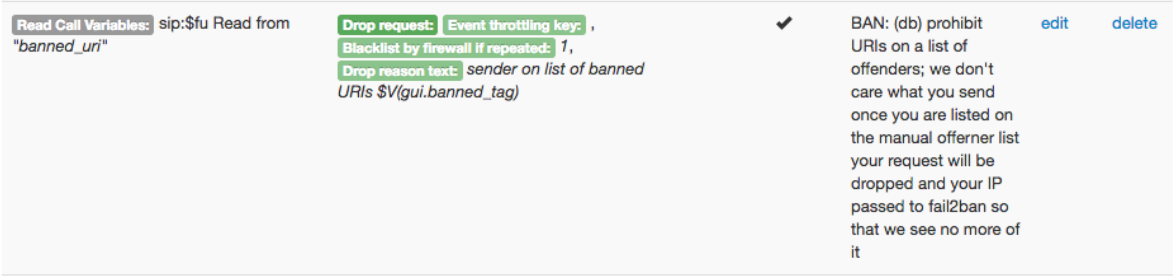


Fig. 35: Rule Example: Prohibited URIs

[1] SIP Vicious: https://github.com/sandrogauci/sipvicious

More Limits

We may also want to constrain the number of parallel calls. We didn't place such a limit in beginning of our rule-set. The reason is that too many call attempts are rejected in the initial part of rule-set and count towards the limit too. When we place the parallel call limit in the rule-base after all unwanted traffic is rejected, the call attempts we chose to decline won't count towards the limit.

Figure *Rule Example: Limit Parallel Calls* shows rule for enforcement of maximum five parallel calls per single IP address. Also note that in this rule, no soft-limit warning is enabled and limit violations add to the security score computed by automated blocking (Section *Automatic IP Address Blocking*).

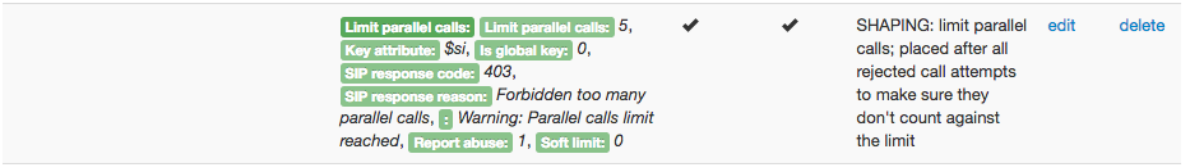


Fig. 36: Rule Example: Limit Parallel Calls

Diagnostic Events

Often SIP messages do appear whose purpose is not entirely clear. Devising a policy that drops them may be premature – they may have some legitimate use which the administrator doesn't understand. It is therefore a good practice to observe them before considering a policy adjustment. This moment of rule processing is perfectly right for this purpose: all traffic that shall be dropped is dropped already.

Example of such is shown in Figure *Rule Example: Report Calls of Unregistered Users*. This rule reports on all non-REGISTER requests for users who have not registered previously. This may be perfectly reasonable for a peering trunk and quite suspicious for a residential user. Gathering these diagnostic events puts an administrator in position to analyze the traffic and create well-targeted policies.

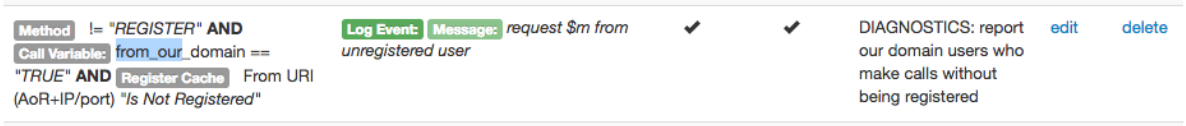


Fig. 37: Rule Example: Report Calls of Unregistered Users

Processing Legitimate Traffic

At this stage of rule processing we have eliminated well-known offending traffic and reported on suspicious traffic. It is time to devise rules that process the traffic considered legitimate: mediation rules, media processing rules, topology hiding, etc. The most important fact for sake of this Section is placement of these rules: they are placed in the very end of a rule-base after all other checks have eliminated unwanted traffic.

Figure *Rule Example: Processing Legitimate Traffic* shows such rules: they implement registration caching and media anchoring to facilitate NAT traversal and off-load the infrastructure behind the ABC SBC . These two rules also contribute to topology hiding: use of media relay hides the actual RTP receivers and registration caching hides the registered contacts.

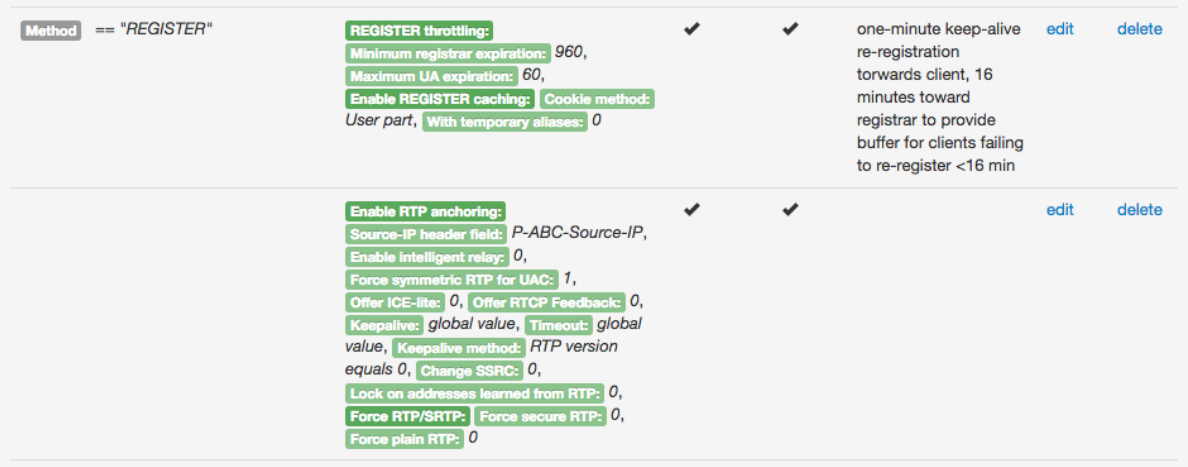


Fig. 38: Rule Example: Processing Legitimate Traffic

Chapter 10

Preview of Experimental Features

This chapter summarizes features that are scheduled to appear in future releases and may, under circumstances, become available earlier in experimental releases. The availability, maturity and scope of the features is subject to change without prior notice. Consult FRAFOS professional services if you wish to inquiry about use of these features.

10.1 Using Two-Factor Authentication for Users

Two-factor authentication (2FA) is a new experimental feature that helps to preserve security of a whole VoIP system even when security of a component is compromised. What sometimes happens is that PBX passwords leak in various ways and stolen passwords are used to make fraudulent calls that appear legitimate. The 2FA system allows to manage "shadow passwords" for SIP users. If a user's account begins to show irregular patterns, identity of the user can be verified using this shadow password. The shadow password is a short string of digits (PIN) which is stored internally at the SBC in parallel to user's SIP credentials.

The system works as follows. On his or her first attempt to make a call, a user is challenged to enroll by submitting a PIN code using DTMF. The user must remember the PIN code for future verification. Subsequent calls work as normally as long as the status of the user doesn't change. The status can be changed manually from "ok" to "soft-limit" by the administrator or an automated tool such as the FRAFOS ABC Monitor. When a user attempts to initiate a call in the "soft-limit" status, he will be challenged to prove his identity using his PIN code. If the user fails to submit the proper PIN code, his status will change to "hard-limit" and further calls will be blocked using an announcement. Otherwise the verification timestamp will be stored and the user will not be prompted anymore for some convenience period of time.

This basic scenario documented below is programmed using ABC rules and can be adjusted to the needs of a specific scenario.

10.1.1 Prerequisites

For the system to work, the following preparations must be made:

- the ABC SBC must be up and running in cloud configuration with a designated configuration master. This allows changes of user status to propagate to all attached SBCs.
- An administrative account and password must be known for use by SBC to manipulate the user status. Ideally a special user is created for this purpose using **"System → Users → Create User"** . In our examples below, we are assuming a user **rpcuser** with password **rpcrpc**.
- the PIN database must be created. To create the database, use **"Tables → Add New Table"** on the configuration master. Make sure you choose **2FA** as the table type.

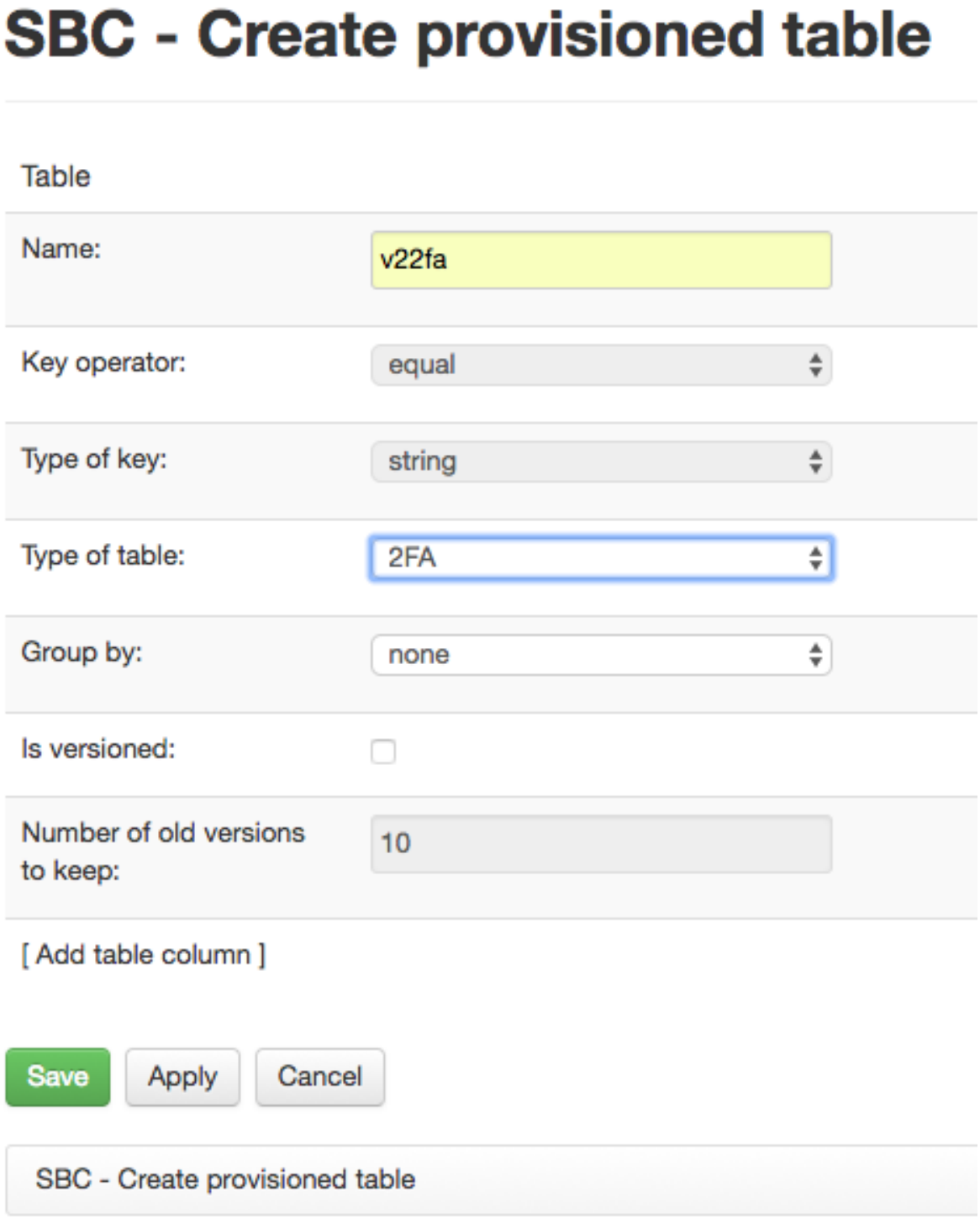


Fig. 1: Two factor authentication: Add a New PIN Table

- a system is required to manipulate user status. This can be done manually by editing the provisioned table, or by this-party tools using xml-rpc, or by deploying an extension to FRAFOS ABC Monitor. An example of XML-RPC use is shown below.

```
#!/usr/bin/python
```

(continues on next page)

(continued from previous page)

```
import xmlrpclib
import ssl
if hasattr(ssl, '_create_unverified_context'):
  ssl._create_default_https_context = ssl._create_unverified_context
# find IP address of config master: grep MASTER /data/sbc/etc/sbc-pullconf.conf
servernew = xmlrpclib.Server('https://rpcuser:rpcrpc@192.168.0.83:1443/rpc.php')
data = {"key_value":"sip:3@abc.com","status":"soft-limit","pin":"1111"};
print servernew.tables.insert_update_rule('twoFA',data);
```

- a loopback CA bound to a loopback interface must be setup. The 2FA is running on a loopback interface so that multiple realms can use the same logic by forwarding traffic to loopback, and the 2FA logic doesn't interfere with the actual realms rules. The following screenshots show how to set up the signaling interface, media interface and the actual CA. The interfaces must use systems physical "lo" interface. There must be also a realm to which the CA belongs.

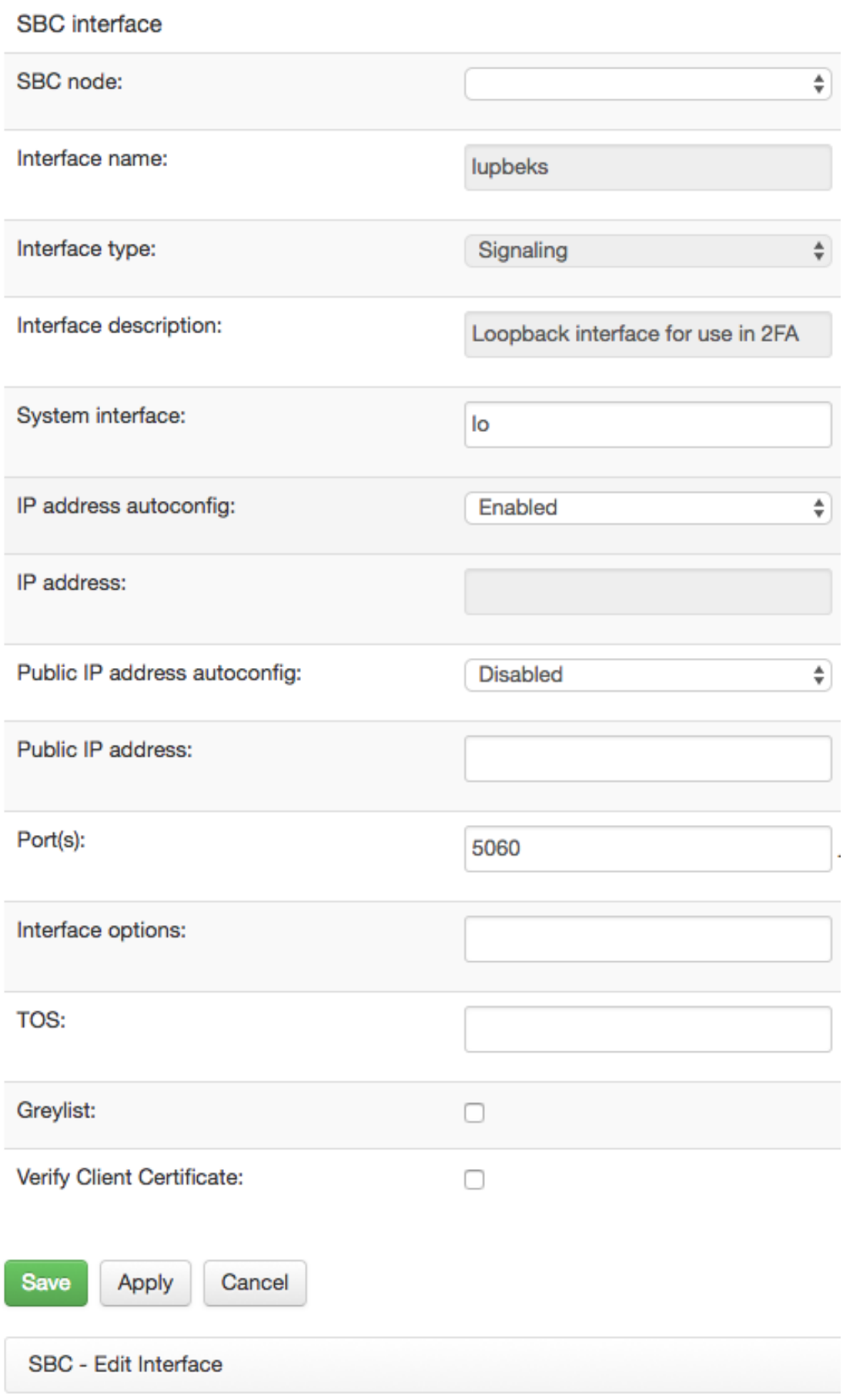


Fig. 2: Loopback signaling interface for two factor authentication

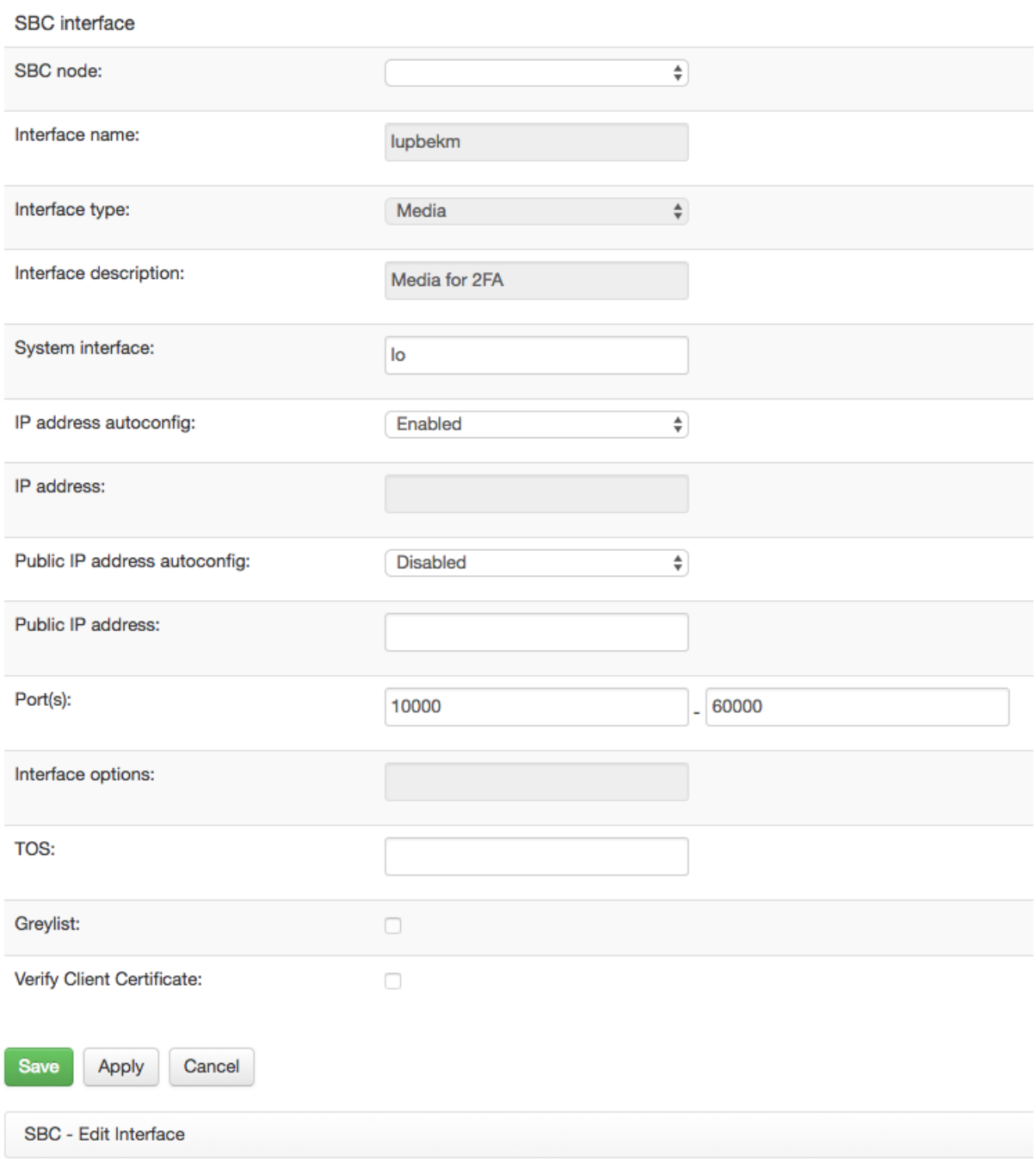


Fig. 3: Loopback media interface for two factor authentication

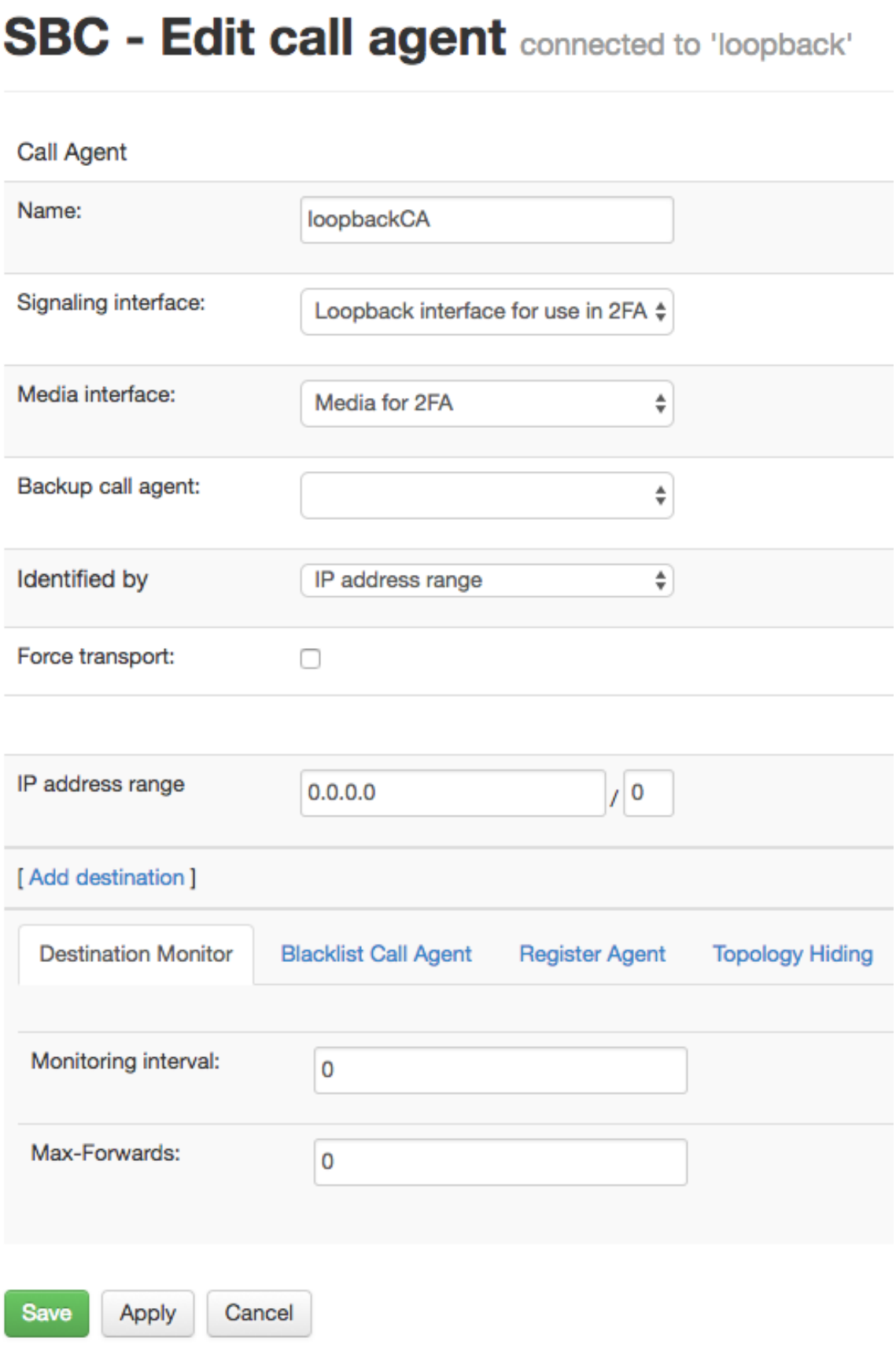


Fig. 4: Loopback Call Agent

10.1.2 Rules for Two Factor Authentication Processing

As mentioned bellow, the actual rules for two factor authentication processing will be tied to a loopback interface. This allows to share the rules for multiple Call Agents, in that the Call Agents forward relevant requests to the loopback interface. This is achieved by rewriting userpart of request URI to indicate the desired action and rewriting the hostpart to the loopback address 127.0.0.1.

The following screenshot shows how the loopback rules are formed. They assume that the user part of the request URI indicates what shall be done with INVITEs for the caller. Whether this is a request from a new user who needs to be enrolled, or a user whose status shall be verified, or a user whose request shall be rejected using a voice announcement.

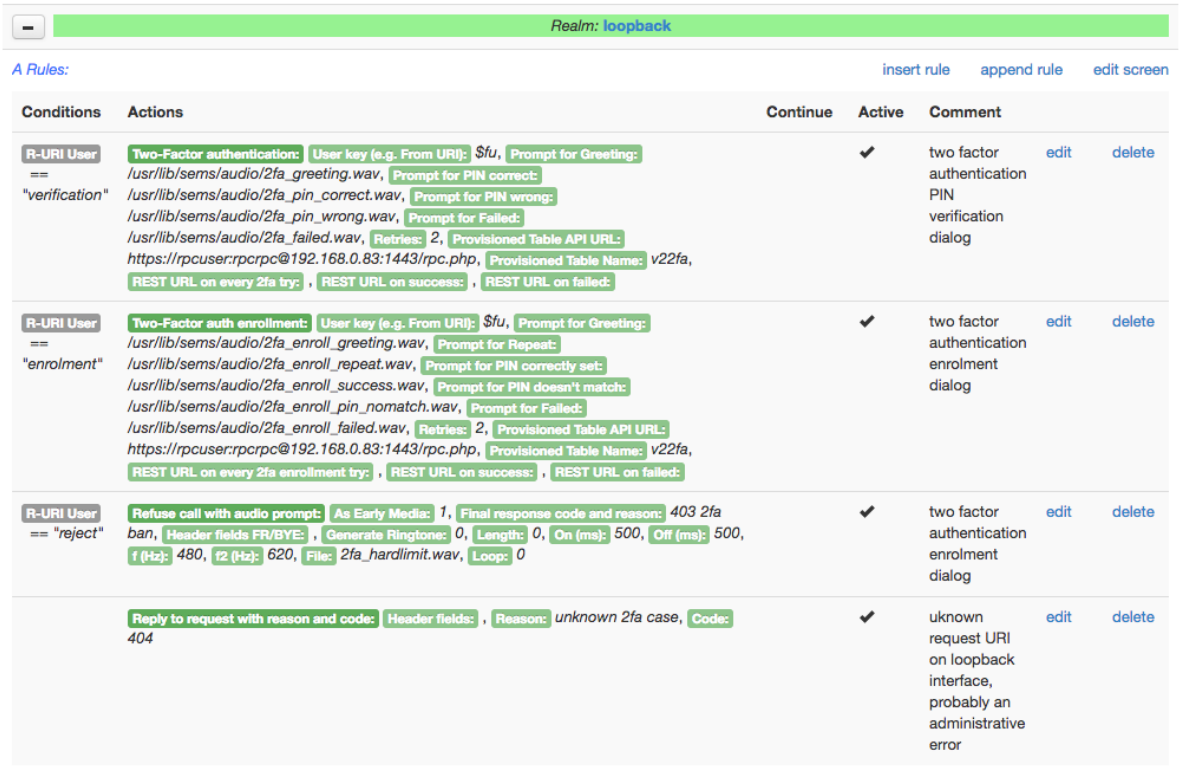


Fig. 5: Loopback Rules

There are two actions: **Two factor authentication** which guides an existing user through a verification dialog. If the user types his proper PIN using DTMF, the action updates user's verification timestamp and the user can continue using the service without further disruptions.

The other action, **Two factor auth-enrollment**, prompts a user to submit his PIN. The PIN is then later used to verify identity of the user.

The rule actions have a couple of parameters. The most important parameter is that of the RPC URI – this is the address of the configuration master and legitimate username and password. Both actions update the PIN database based on their completion status. The audio filenames can be changed for a different user experience. RESTful URIs can be optionally used to notify external applications when a rule action finishes with success or failure.

The parameter "Source IP" can be used to set the remote IP address which is recorded with the two factor authentication record in the database. As the processing is executed after being looped on a loopback interface, the real remote IP may be passed on, e.g. by adding a header 'P-ABC-Source-IP' with the correct remote IP. That can be used here with the value '$H(P-ABC-Source-IP)'.

10.1.3 Rules for determining User Status and discriminating by it

The more sophisticated part of the rules discriminates how to treat respective users. An example of such is shown in the following screenshot.

The very first rule retrieves the user status from the current database. The table attributes, such as PIN and status, are then available for processing as call variables.

The first condition selects users whose status is set to “hard-limit”. In that case, incoming INVITEs are forwarded to the loopback interface with “reject” in userpart of request URI, and rejected from there.

The next rule targets users for whom no records are available. Their INVITEs are forwarded to loopback for enrolment.

Subsequent conditions try to determine whether the user is in the **soft-limit** status, and how recently he has verified his identity. If the last verification is too long ago (24 hours in this example), the INVITE is forwarded to the loopback interface for PIN verification.

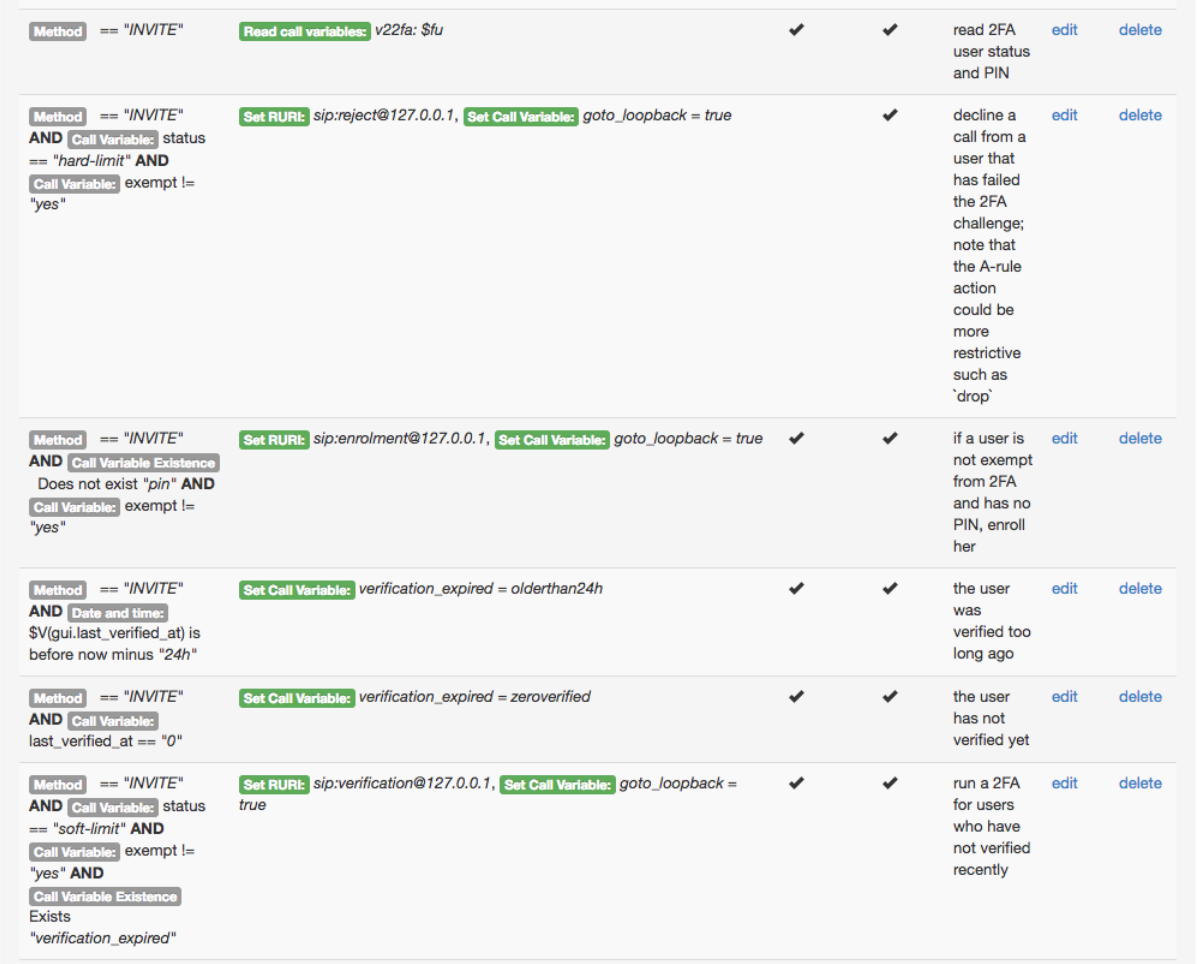


Fig. 6: User Discrimination in Two factor authentication Ruleset

If none of these conditions matched, the rule processing continues “as usual”. That’s the case when user status is “ok” or if the user’s identity was verified recently.

10.1.4 Routing Rule to Connect Two Factor Authentication Processing and User Discrimination

There is one remaining piece to connect the user discrimination and 2FA processing rules: a routing rule. The user discrimination rules have already set the loopback address in request URI and defined a variable "goto_loopback". We still need to act upon these. This is fortunately easy to set up:

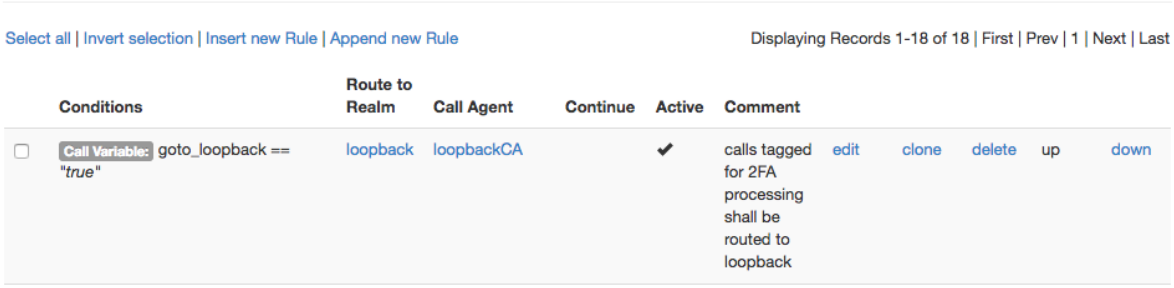


Fig. 7: Two Factor Authentication Routing Rule

10.1.5 Scenario Modifications

The two-factor authentication scenario can be modified in many different ways using the ABC rules. The period for which a user doesn't have to be re-validated may be extended or shortened. The IP address of a request can be checked against the IP address from which the identify was verified the last time (last_verified_from_ip) – then a successful verification only validates calls from the same IP address. The way calls from a user in *hard-limit* status are declined can be changed. Note however, that the 2FA application is still in experimental status and untested scenarios may or may not work as expected.

10.2 AWS: Reputation Lists

Monitor-steered firewalling allows to combine Monitor intelligence about misbehaving users and ABC SBC's capability to filter out their traffic before it causes harm. It is based on the notion of reputation list and works as follows: an ABC Monitor collects events from all associated SBCs as usual. It uses its data-streaming logic to identify misbehaving traffic sources and posts such to a reputation list. SBCs are subscribed to the list, receive a notification when a new IP address is published by the ABC Monitor, and block the source then.

Use of the ABC Monitor to decide which IP addresses to block is particularly advantageous for several reasons:

- the ABC Monitor collects big data about users and their behaviour and is therefore in a very good position to make sophisticated decisions which IP addresses should be blocked.
- the centralized nature of the ABC Monitor allows to convey problematic IP address to all managed SBCs as soon as any of them detects them
- the ABC Monitor has a global view of multiple ABC SBC and can identify misbehaving traffic sources even when they spread their traffic to remain low profile on each managed SBC but their total traffic is beyond a critical mass.

Currently, the reputation list is facilitated using AWS Simple Notification Service (SNS). It is not necessary for the ABC Monitor and ABC SBC to run on AWS but both must have access to AWS SNS service.

10.2.1 Setting Up ABC SBC for Use of Reputation List on AWS

Before you begin, the following prerequisites must be set up in AWS and in the ABC SBC configuration:

- In AWS, there must be an SNS topic, to which the Monitor is allowed to write and from which the ABC SBC is allowed to read.
- AWS Identity must be properly configured on the ABC SBC under **"Global Config → AWS"**. Set AWS region for the SNS, and key id and secret key for identity that can subscribe to the SNS topic. We recommend that you set up a special IAM user with privileges limited to receiving SNSs for this purpose.
- On the ABC SBC, an XMI interface must be properly set up and run on an IP address reachable by SNS. Private IP address not connected to AWS via a VPN will be unreachable for the notifications and the subscription will fail. If the ABC SBC is running on AWS, the option "Public IP address autoconfig" must be therefore set to "Amazon Method".
- RestFul interface for processing notifications must be enabled on the XMI interface. To do so, choose the XMI interface name under **"Global Config → Misc → RESTful interface XMI name "**. Make sure that the port number under **"Global Config → Misc → RESTful interface XMI port"** is open in the SBC's security group.

To subscribe to the SNS, find **"System → Firewall → Subscription to AWS Notification Service"**, click **"Subscribe"** and include the SNS topic's ARN. After the subscription is successfully completed, the IP addresses learned from the reputation list appear under "External FW blacklist".

10.2.2 Setting Up ABC Monitor for Use of Reputation List on AWS

- In AWS, there must be an SNS topic, to which the Monitor is allowed to write and from which the ABC SBC is allowed to read.
- ABC Monitor instance must be assigned a proper IAM role to publish SNS messages.
- ABC Monitor instance must be configured to post to the topic identified by its region and ARN. The settings are under **"Settings"**. In this configuration section, it is also possible to set threshold for Exceeded Limits that may eventually cause a source address to be published on a reputation list.
- The administrator must choose which type of Exceeded Limits will place a source address on the reputation list. To do so turn on/off checkboxes under **"SNS Settings"**.

10.3 Server Transaction limits

The server transaction limits allows for limiting the number of running SIP server transactions (UAS transactions). This mechanism, when properly configured, offers a very effective protection against burst of new transactions typical for denial of service attacks.

These limits will allow the administrator to be warned (events are generated toward the ABC Monitor) when certain limits are passed (soft limits) and limits to be enforced by rejecting new transactions (hard limits) with `503 Overloaded`. In case requests are actively rejected, ABC Monitor events are sent as well.

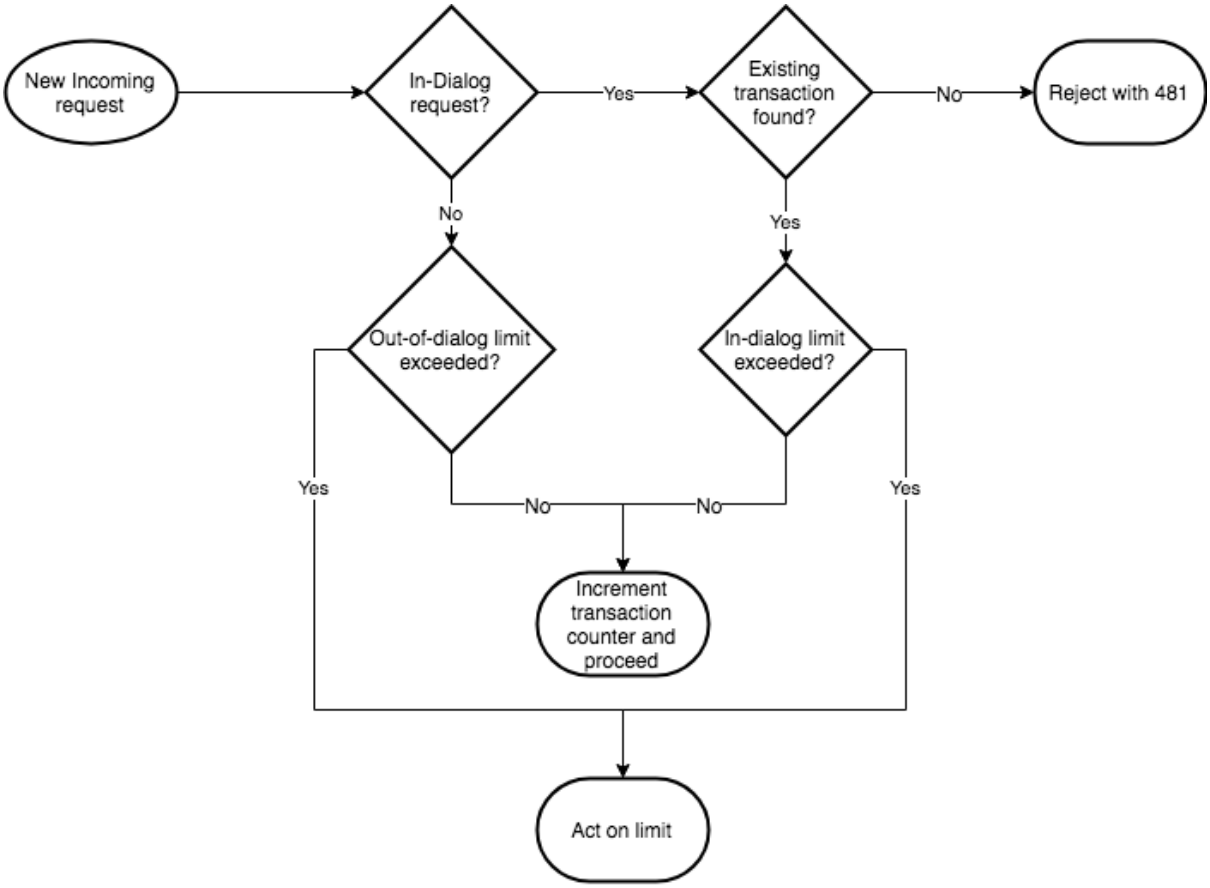


Fig. 8: Transaction limit flow diagram

The diagram above shows the behaviour related to transaction limits when a new request (not a retransmission) is received. The action taken on a limit breach depends on the type of limit (soft vs. hard limit), as described above.

It is important to note that the very same transaction counter is used to check both types of limits (in-dialog and out-of-dialog), so that the in-dialog limits must necessarily be higher than the out-of-dialog limits. The difference between both is the guaranteed number of in-dialog transactions that can be held.

Soft limit for out-of-dialog transactions (event logging only, use 0 to disable):	0
Hard limit for out-of-dialog transactions (event + reply 503, use 0 to disable):	0
Event throttling for soft/hard OOD limit (in seconds, use 0 to disable):	0
Soft limit for in-dialog transactions (event logging only, use 0 to disable):	0
Hard limit for in-dialog transactions (event + reply 503, use 0 to disable):	0
Event throttling for soft/hard DLG limit (in seconds, use 0 to disable):	0

Fig. 9: Transaction limit settings

The transaction limit settings can be found in the global config parameters, in the category “SEMS”.

10.3.1 Setting proper limits

The easiest method for setting proper limits is to monitor the number of UAS transactions while the SBC is operating under normal conditions with the ABC Monitor and to apply a factor of at least 2 to these numbers before setting limits.

For any of the transaction limits fields, a value of 0 means that the limit is deactivated.

In order to start using the limits without impairing production traffic, the soft limits should be set first, together with the event throttling to avoid generating too many events.

Once the soft limit give satisfactory results, meaning that events are generated only on significant load peak, the hard limits can be with a safe margin (at least +30%).

The in-dialog limits should be set very carefully, as it impacts greatly the stability of the system. In particular, BYE requests could be lost in case the in-dialog transaction limit is set improperly.

10.4 Advanced Load Balancing

This is work in progress. Show: - when we need advanced load-balancing (consistent) - how to set up the rules and config values, possibly what to do CLI - CLI troubleshooting

10.5 New restify CDR process

For information about the legacy behavior, please refer to *Call Data Records (CDRs)*.

Since ABC SBC 4.5, the new CDR-ng feature may be enabled in ABC cluster manager under *Global config* > *CDRs* > *Enable new version of CDRs (CDR-NG)*. The new process monitors event from a target redis list. If a event type is matched with one from the watch list (*call-end* or *conf-leave* for example), a CDR is generated in a csv format. The csv content is based on the event fields, in a format specific to the event type. The CDR is then forwarded to a specific syslog facility.

Please note that by default, *syslog-ng* is configured to redirect a process's messages from a facility to a target file. ABC SBC default configuration for CDR is the following :

Process	Syslog facility	Target file
restify-cdr	LOCAL1	/var/log/frafos/cdr/cdrNG.log
restify-cdr	LOCAL2	/var/log/frafos/cdr/cdrNGconf.log

The configuration file is located in */etc/frafos/restify-cdr.conf*. The template (*/etc/frafos/templates/restify-cdr/restify-cdr.conf.tmpl*) may be overloaded, as described in *Command Line Reference*.

By default :

- *call-end*, *call-attempt* and *conf-leave* event are watched
- *call-end* and *call-attempt* CDR are forwarded to the *LOCAL1* facility
- *conf-leave* CDR are forwarded to the *LOCAL2* facility

The following formats are defined by default :

- *classic*: 1-1 call CDRs (*call-end*, *call-attempt*)
- *webconf*: web conference based CDRs (*conf-leave*)

Chapter 11

Reference of Actions

The actions are grouped as follows:

- *SIP Mediation*
- *SDP Mediation*
- *Monitoring and Logging*
- *Traffic Shaping*
- *Media Processing*
- *SIP Dropping*
- *Scripting*
- *Register Processing*
- *External Interaction*
- *NAT Handling*
- *Other*

11.1 SIP Mediation

Action Name	Description	Parameters
Set RURI	Set request URI to a new value	• new URI
See *Request-URI Modifications*.		
Prefix RURI user	Prefix userpart of request URI	• prefix string
See *Request-URI Modifications*.		
Set RURI user	Replace userpart of request URI	• new userpart
See *Request-URI Modifications*.		
Append to RURI user	Add a suffix to userpart of request URI. The result is accumulated if actions is used multiple times.	• suffix
See *Request-URI Modifications*.		

continues on next page

Table 1 – continued from previous page

Action Name	Description	Parameters
Strip RURI User	Remove leading characters of userpart of request URI	• number of leading characters
See *Request-URI Modifications*.		
Set RURI Host	Replace hostpart of request URI	• new hostpart
	See *Request-URI Modifications*.	
Set RURI Parameter	Set request URI parameter	• parameter name • parameter value
See *Request-URI Modifications*.		
Set From	Replace From Header Field Value	• From HF value
See *Changing Identity*.		
Set From display name	Replace From Display name	• new From Display name
See *Changing Identity*.		
Set From User	Replace userpart of From URI	• new From userpart
See *Changing Identity*.		
Set From Host	Replace hostname of From URI	• new From hostname
See *Changing Identity*.		
Set To	Replace To Header Field Value	• To HF value
See *Changing Identity*.		
Set To Display Name	Replace To Display name	• new To Display name
See *Changing Identity*.		
Set To User	Replace userpart of To URI	• new To userpart
See *Changing Identity*.		
Set To Host	Replace hostname of To URI	• new To hostname
See *Changing Identity*.		
UAC auth	Authenticate on behalf of UAC against a UAS. Any request passing this action and challenged to authenticate by a downstream server will be resent with credentials passed in the action's parameters.	• username • password • realm
See *Changing Identity*.		

continues on next page

Table 1 – continued from previous page

Action Name	Description	Parameters
UAS auth	Authenticate a UAC against the SBC. Either HA1 or password can be provisioned on the SBC; HA1 is safer as the plaintext password does not need to be saved on the SBC. The HA1 can be calculated as MD5(username:realm:password) or with the tool sbc-calc-ha1 on the command line. Can be used together with provisioned tables and the "Save REGISTER contact in registrar" action to create a full registrar.	• username • realm • H(A1) or password
Remove Header	Removes all occurrences of a header field. The action is applied to initial message, newly added header fields are not removed	• header field name
See *SIP Header Processing*.		
Add Header	Add a new Header Field to a request	• HF Name • HF Value
Replace header value (on leg)	Same as "Replace header value" but acts on messages on call leg only. E.g. putting a rule on A rules of CA1: *[CA1] INVITE -> [SBC] -> [CA2] 200 OK -> [SBC] rule-applied -> [CA1]* E.g. putting a rule on C rules of CA2: *[CA1] INVITE -> [SBC] rule-applied - > [CA2] 200 OK -> [SBC] -> [CA1]*	• header name • search • replace with
Insert or Replace header (on leg)	Tries to insert a header field to a request. Unless "replace existing" is enabled, the insertion will NOT take place if a header with the same name exists. If "replace existing" is enabled, the header is replaced with the given value. "Header value" may be a call variable value (ex: *$V(gui.fullname)*) or back-reference.	• header name • header value • replace existing
See *SIP Header Processing*.		
Set header whitelist	Removes all but mandatory and white-listed header-fields. Names are comma-separated, case-insensitive and need to specify compact forms explicitly. The list is applied to the final appearance of the INVITE request after all A and R rules have been processed.	• comma-separated header-field name list

continues on next page

Table 1 – continued from previous page

Action Name	Description	Parameters
See *SIP Header Processing*.		
Set header blacklist	Removes all blacklisted header-fields. Names are comma-separated, case-insensitive and need to specify compact form explicitely The list is applied to the final appearance of the INVITE request after all A and R rules have been processed.	• comma-separated header-field name list
See *SIP Header Processing*.		
Update Supported header	Allows simplified manipulation with Supported header field content (since version 4.5).	• operator (Add / Remove / Set tags) • comma-separated list of option tags
See *Option tags*.		
Update Require header	Allows simplified manipulation with Require header field content (since version 4.5).	• operator (Add / Remove / Set tags) • comma-separated list of option tags
Update Allow header	Allows simplified manipulation with Allow header field content. Note that "Add" operator will not add unless Allow header already exists, set via "Set" operator or "Default tags" are specified.	• operator (Add / Remove / Set tags) • comma-separated list of option tags • Direction • Apply on • Default tags
Replace URI header user	Allows modifying "user" part on headers containing a URI. I.e. Refer-to: sip:USER@host	• Header name • Search • Replace with
Replace URI header host	Allows modifying "host:port" part headers containing a URI. I.e. Refer-to: sip:user@HOST:PORT	• Header name • Search • Replace with
Replace headers of URI header	Allows modifying headers in headers containing URIs. I.e. Call-ID in Refer-to: <sip:user@host?Call-ID= 55432%40alicepc.atlanta.example.com> can be manipulated with "header name = refer-to", "name of the header in URI = call-id", "Search = 432@alice", "replace with = 433@bob".	• Header name • Name of the header in URI • Search • Replace with

continues on next page

Table 1 – continued from previous page

Action Name	Description	Parameters
Insert or replace headers of URI header	Allows modifying headers in URI of headers containing a URI. I.e. NEW-hdr in Refer-to: <sip:user@host?Call-ID=55 432%40alicepc.atlanta.example.com& NEW-hdr=value> can be added with this.	• Header to modify • Header name • Header value • Replace if exists
See *Option tags*.		
Add Dialog Contact Parameter	Add parameters to the Contact URI generated by the SBC	• Leg: A or B • parameter name • parameter value
See *Other mediation actions*.		
Set Contact-HF parameter whitelist/blacklist	Specify which Contact header field parameters in incoming request to forward downstream.	• comma-separated list of parameter names
Set Contact-URI parameter whitelist/blacklist	Specify which Contact URI parameters in incoming request to forward downstream.	• comma-separated list of parameter names
See *Other mediation actions*.		
Forward Contact-HF parameters	Forward all Contact header field parameters “as is” downstream.	• none
Forward Contact-URI parameters	Forward all Contact URI parameters “as is” downstream.	• none
See *Other mediation actions*.		
Translate Reply Code	Translate SIP reply codes to other value	• matching reply code • new reply code • new reason phrase
See *Other mediation actions*.		
Set Max Forwards	Reset the number of hops a request can be forwarded to specified value	• the new value of Max-Forwards header field
See *Other mediation actions*.		
Enable transparent dialog IDs	Enforce use of the same dialog IDs on both sides of a call	• none
See *Other mediation actions*.		
Forward Via-HFs	Force the SBC to keep the Via header fields while forwarding the request.	• none
See *Other mediation actions*.		
Diversion to History-Info	converts SIP diversion header-field into History Info	• none
See *Other mediation actions*.		

continues on next page

Table 1 – continued from previous page

Action Name	Description	Parameters
Call transfer handling	Defines the mode in which REFERs are handled: rejection, local processing or forwarding. Option to reconnect if transfer ends in 4xx during unattended transfer. Option to not terminate referrer leg when the unattended transfer completes.	• REFER processing mode • Reconnect on failure • Do not terminate • after transfer
See *Other mediation actions*.		
Handle INVITE with Replaces header	Activates internal processing of INVITE with Replaces header	• none
See *INVITE with Replaces handling*.		
Map Replaces header	Activates mapping of dialog identifiers in INVITE with Replaces	• none
See *Mapping Dialog-IDs in INVITEs with Replaces*.		
Set Content Type whitelists/blacklists	Specifies which SIP payload types (such as SDP) will be permitted.	• comma-separated list of content types
See *Other mediation actions*.		
Enable SIP Session Timer caller-leg	Enforce use of Session timer	• session expiration (sec) • minimum expiration (sec) • let remote refresh
See *Controlling SIP Session Timers (SST)*.		
Enable SIP Session Timer callee-leg	Enforce use of Session timer	• session expiration (sec) • minimum expiration (sec) • let remote refresh
See *Controlling SIP Session Timers (SST)*.		
Add X-Org-ConnID header	The X-Org-ConnID header field contains a unique value that remains constant for the duration of the transaction and any dialog created from this request. By enabling this action, a X-Org-ConnID header is added to every outgoing initial SIP INVITE request product of this dialog. The header helps to correlate calls that have been internally redirected (due to a 302 SIP response) or blindly transferred (due to a REFER SIP request). The value can be retrieved in the CDR by specifying the keyword "$x_org_connid" in the cdr_format (see cc_syslog_cdr.conf).	

11.2 SDP Mediation

Table 2: SDP Mediation Actions

Action Name	Description	Parameters
Set CODEC Whitelist	Remove all but listed codecs from SDP.	• comma-separated codec-list case insensitive
See *CODEC Filtering*.		
Set CODEC Blacklist	Remove all listed codecs from SDP.	• comma-separated codec-list, case insensitive
See *CODEC Filtering*.		
Set CODEC Preferences	Define the order in which available codecs are chosen.	• comma-separated codec-list
See *CODEC Preference*.		
Set SDP attribute whitelist/blacklist	Removes specified CODEC attributes from SDP payload.	• comma-separated list of attribute names
See *CODEC Preference*.		
Set Media whitelist	Permit only listed media types, audio or video	• audio,video list
See *Media Type Filtering*.		
Set Media blacklist	Remove listed media types, audio or video	• audio,video list
See *Media Type Filtering*.		
Drop early media	Drop early media (audio only).	• none
See *Early Media, Ring Back Tone and Forking*.		
Drop SDP from 1xx replies	Drop SDP from listed 1xx replies	• list of affected reply codes
See *Early Media, Ring Back Tone and Forking*.		
Insert or Replace SDP Session Attribute (on leg)	Try to insert a session-level attribute to all requests/replies on call leg. Unless "replace with" is enabled, the insertion will not take place if an attribute with same name exists. If replace with is enabled the value of the atribute with the same name is changed to "Attribute value". Attribute name is the name to replace , supports replacements. Attribute value supports replacements & back-references. Replace with replaces if already exists.	• Attribute name • Attribute value • Replace with

continues on next page

Table 2 – continued from previous page

Action Name	Description	Parameters
Replace SDP Session Attribute (on leg)	Replace an SDP session attribute on all requests/replies on a call leg. Attribute name is the name to replace , supports replacements. Search is a regexp to match the part to be replaced. Replace with holds the value to be replaced with, supporting replacements & back-references.	• Attribute name • Search • Replace with
Insert or Replace SDP Media Attribute (on leg)	Try to insert a media-level attribute to all requests/replies on call leg. Unless "replace with" is enabled, the insertion will not take place if an attribute with same name exists. If replace with is enabled the value of the atribute with the same name is changed to "Attribute value". Attribute name is the name to replace , supports replacements. Media is a regexp matched against the *m=* media lines to select specific ones. Supports replacements & back-references. Attribute value supports replacements & back-references. Replace with replaces if already exists.	• Attribute name • Media • Attribute value • Replace with
Replace SDP Media Attribute (on leg)	Replace an SDP media attribute on all requests/replies on a call leg. Attribute name is the name to replace , supports replacements. Media is a regexp matched against the *m=* media lines to select specific ones. Supports replacements & back-references. Search is a regexp to match the part to be replaced. Replace with holds the value to be replaced with, supporting replacements & back-references. This action can be used for payload id re-mapping if used with RTP anchor. E.g. attr. name, media, search, replace with values *rtpmap*, *.**, *^98 XYZ*, *105 XYZ* respectively will replace payload id 98 with 105 in relayed RTP packets.	• Attribute name • Media • Search • Replace with

continues on next page

Table 2 – continued from previous page

Action Name	Description	Parameters
Remove SDP Media Attribute (on leg)	Remove an SDP media attribute on all requests/replies on a call leg. Attribute name is the name to remove, supports replacements. Media is a regexp matched against the *m=* media lines to select specific ones. Supports replacements & back-references. Search is a regexp to match the line to be removed. I.e. in removal of payload with id 102 m=audio 8012 RTP/AVP 102 103 a=rtpmap:102 telephone-event/48000 a=rtpmap:103 telephone-event/8000 "Attribute name" would be rtpmap, "Media" would be compared against "audio 8012 RTP/AVP 102", "Search" would be compared to "102 telephone-event/48000", and would result in m=audio 8012 RTP/AVP 103 a=rtpmap:103 telephone-event/8000	• Attribute name • Media • Search
Insert or Replace SDP Payload Attribute (on leg)	Try to insert a payload-level attribute to all requests/replies on call leg. Unless "replace with" is enabled, the insertion will not take place if an attribute with same name exists. If replace with is enabled the value of the atribute with the same name is changed to "Attribute value". Attribute name is the name to replace , supports replacements. Media is a regexp matched against the *m=* media lines to select specific ones. Supports replacements & back-references. Codec is a regexp matched against the respective *rtpmap=xyz <CODEC>*. Supports replacements & back-references. Attribute value supports replacements & back-references. I.e. for *fmtp*, it is placed as *fmtp: xyz <VALUE>*. Replace with replaces if already exists.	• Attribute name • Media • Codec • Attribute value • Replace with

continues on next page

Table 2 – continued from previous page

Action Name	Description	Parameters
Replace SDP Payload Attribute (on leg)	Replace an SDP payload attribute on all requests/replies on a call leg. Parameters are almost the same as Insert or Replace SDP Payload Attribute Action. Search is a regexp to match the part to be replaced. I.e. for *fmtp* it is compared against *fmpt:xyz <SEARCH>*.	• Attribute name • Media • Codec • Search • Replace with
Limit telephony event list (on leg)	Limit telephony events attribute on all requests/replies on a call leg. Media is a regexp matched against the *m=* media lines to select specific ones. Supports replacements & back-references. Telephony events is a list such as *0-16,66* that will filter out anything that is not in it.	• Media • Telephony events

11.3 Monitoring and Logging

Action Name	Description	Parameters
Increment SNMP counter	Increment an SNMP counter	• counter name • increment
See *User Defined Counters*.		
Log received traffic	Log SIP/RTP traffic concealed with logging into PCAP file. The general log level is used if none is set for that call.	• log type • PCAP file name Use filename with .pcap extension.
See *Diagnostics Dashboard*.		
Log Event	Generate custom event	• event text
See *SEMS Parameters*.		
Set log level	Set a specific log level for this traffic. Note: The global log level will be applied until this Action is processed.	• log level (see Section *Reference of Log Level Parameters*)
See *Diagnostics Events*.		
Log Message	Use syslog facility	• log level • message text
Log Message for Replies	Report on a transaction that completed with a specific response code. Depending on parameters, such a report can lead to blacklisting or promoting a whitelisted IP address. Typically used to alarm on requests that were declined because of a possible security risk. The action can report via events, syslog or suggest that the request originator is put on blacklist or promoted on a greylist.	• reply codes that trigger the reports (comma-separated list or asterisk for any response code) • syslog level • use syslog • send an event • Blacklist UAC IP Address • Blacklist UAS IP Address • Greylist UAC IP Address • Greylist UAS IP Address
See *Automatic IP Address Blocking* and *Automatic Proactive Blocking: Greylisting*.		
Log to grey list	Promote a source IP address from greylist to whitelist.	• label – token that differentiates internally the promotion reason; choose some short descriptive string
See *Automatic Proactive Blocking: Greylisting*		

11.4 Traffic Shaping

Action Name	Description	Parameters
Limit parallel calls	Put a quota on number of parallel calls for some specific part of traffic identified by a key. The limit applies separately to inbound and outbound traffic in A and C rules respectively and realm or CA to which the action's rule is linked unless "global key" is turned on. Exceeding calls attempts are rejected using 403.	• max number of calls • key (optional) that identifies a subset traffic • global key • SIP header • soft limit • report abuse • SIP response code and phrase
See *Traffic Limiting and Shaping*.		
Limit CAPS	Put a quota on number of call attempts per second for a traffic subset identified by a key. The limit applies separately to inbound and outbound traffic in A and C rules respectively and realm or CA to which the action's rule is linked unless "global key" is turned on. Authentication counts towards the limit as well. Exceeding calls attempts are rejected using 403.	• max number of request per unit of time • time unit – length in seconds • key and global key • SIP response code and phrase • report abuse • soft limit
See *Traffic Limiting and Shaping*.		
Limit Bandwidth per Call	Put a quota on RTP traffic in kbps. A rules steer bandwidth for inbound calls, C rules for outbound. Exceeding RTP traffic is dropped.	• limit (kbps) • key and global key • SIP response code and phrase • soft limit • report abuse
See *Traffic Limiting and Shaping*.		
Limit Bandwidth	Don't admit signaling if its codecs in SDP exceed a limit.	• limit (kbps)
See *Traffic Limiting and Shaping*.		
Set call Timer	Terminate a call if it exceeds a limit length.	• max call length (sec)
See *Setting Call Length Limits*		

11.5 Media Processing

Action Name	Description	Parameters
Enable RTP anchoring	Anchors RTP media to the ABC SBC. Allows to centralize media forwarding. Additionally, ICE connectivity checks and RTP keepalive can be introduced for anchored calls. If RTP timeout is introduced and no RTP packet appears, the call is terminated. Anchoring is a prerequisite for other media processing such as recording. RTCP report generation can also be configured to happen on certain conditions described in "RTCP Gen.". RTP Gen. "Always" disables RTCP relay and sends the generated RTCP.	• Force symmetric media • Enable intelligent relay (IR) • Source IP Header field for IR • Offer ICE-lite • Offer RTCP feedback • Keepalive (sec) • timeout (sec) • Ignore ICE Offer • RTCP Generation • RTCP Interval
See *Media Anchoring (RTP Relay)*		
Force RTP/SRTP	Enforces conversion to the requested protocol in C-rules. In A-rules it only admits specified protocol and declines requests otherwise.	• Key exchange mechanism (DTLS/SDES)
See *RTP and SRTP Interworking*		
Activate audio recording	Record audio into stereo WAV file. *call-end* events will contain a link to the file holding the recording. The link will be indexed by the *audio_file* field.	• destination
See *Audio Recording*		
Activate transcoding	Activate transcoding for list of codecs. Listed codecs are added to SDP and transcoded if selected.	• comma-separated codec list
See *Transcoding*		
Relay DTMF as AVT RTP	Relay DTMF as AVT RTP packets (RFC4733/RFC2833) Direction parameter sets on which direction to apply the conversion on. E.g. setting it to "To B leg" on C rules would apply the conversion on DTMF generated by A leg (caller). Direction defaults to "To B leg" and "To A leg" in A and C rules respectively.	• Direction
See *Other mediation actions*.		
Relay DTMF as SIP INFO	Relay DTMF as SIP INFO	• none
See *Other mediation actions*.		
Refuse call with audio prompt	see *SIP Dropping* bellow	
Join meet-me conference	Make a call join a conference.	• Enter room via keypad • Room • Minimal room length • Unacceptable rooms • Room prefix • Split Room number and participant ID • Position to split room - Room is PIN protected • PIN

Action Name	Description	Parameters
Play prompt on final response	Play an audio announcement on receipt of a negative final response from downstream	• SIP response codes to trigger the announcement • As Early Media • New response code if "as early media" • Optional header fields • announcement WAV filename OR ... • ... characteristics of a generated ringtone
See *Playing Audio Announcements*.		
Generate Ring-Back Tone	Play an audio file or a dual-frequency tone instead of default ringing tone.	• on downstream 180: start playing when a 180 response arrives • on Timer: start playing if a number of seconds elapses (turned off if zero) • Generate Ringtone if turned on, a dual-tone with specified frequencies and durations will be played; otherwise a specified audio file will be used. • Loop: when audio file is chosen this option chooses whether to play it once or in a loop
See *Playing Audio Announcements*.		
Activate Music On Hold	Use this action on a call to play an audio file when a call participant puts the call on hold. It is possible to specify how to signal the onhold status in SDP.	• music file name • playback in loop • Hold indication (sendonly, sendrcv, preserve incoming, inactive, rfc2543 0.0.0.0 IP
See *Playing Audio Announcements*.		
Activate Inband DTMF Detection	Use this action together to enable detection of inband DTMF in relayed RTP packets. Direction can be used to set which direction the detection will be enabled. Mode can be used to i.e. not enable the detection if RTP/AVT is supported in the SDP.	• Direction • Mode

11.6 SIP Dropping

Action Name	Description	Parameters
Reply to request	Send a negative response to a SIP request	• Code • Reason phrase • optional header field
See *Manual SIP Traffic Blocking*.		
Drop request	Drop request silently	• Event throttling key
See *Manual SIP Traffic Blocking*.		
Allow unsolicited NOTIFYs	Allow forwarding NOTIFY requests without a prior subscription (either implicit with REFER, or explicit with SUBSCRIBE).	• none
See *Other mediation actions*.		

11.7 Scripting

Action Name	Description	Parameters
Set Call Variable	Stores a computing result in an variable. The variable can be tested using the Call Variable condition and/ or referred to from actions using the $V(gui.varname) replacement.	• variable name • variable value
See *Binding Rules together with Call Variables*.		

11.8 Register Processing

Action Name	Description	Parameters
Enable REGISTER caching	Stores a cached copy of REGISTER contacts before forwarding.	• none
See *Registration Handling Configuration Options*.		
Retarget R-URI from cache	Rewrites AoR in request URI with contacts cached using Enable REGISTER caching	• enable NAT handling • enable sticky transport
See *Registration Handling Configuration Options*.		
REGISTER throttling	Force UAs to refresh registrations within a time window. Particularly useful to trigger REGISTER-based keep-alives to facilitate NAT traversal.	• minimum registrar expiration • maximum UA expiration
See *Registration Handling Configuration Options*.		
Save REGISTER contact	Act as local registrar and store registers locally.	• none
See *Registration Handling Configuration Options*.		
Restore contract from registrar	Restore contact from registrar	• none
See *Registration Handling Configuration Options*.		

11.9 External Interaction

Action Name	Description	Parameters
ENUM query	make an ENUM dip. The queried value may contain replacement expression, suffix is appended to the query.	• queried value • domain suffix • ENUM services
See *ENUM Queries*.		
Read call variables over REST	Do REST query to given URL and set call variables received in reply.	• REST URI
See *RESTful Interface*.		
Read call variables from table	Read variables from a provisioned table	• table name • query key
See *Provisioned Tables*.		

11.10 NAT Handling

Action Name	Description	Parameters
Enable dialog NAT handling.	Remember during dialog lifetime where the initial dialog-initiating request came from and sends all subsequent SIP traffic there.	• none
See *NAT Traversal*.		

11.11 Other

Action Name	Description	Parameters
Support serial forking proxy	Permit to reset early media upon 181-indicated serial forking	• none
See *Early Media, Ring Back Tone and Forking*.		
Fork	Fork a new parallel branch to a URI	• SIP URI
See *Early Media, Ring Back Tone and Forking*.		

11.12 Default Audio Files

Most of the prompt's sample rate is 8000. It isn't necessarily required, as `sems` resample them. Note that wideband samples may sounds nicer.

11.12.1 Meet-me set PIN audio prompts

The meet-me set PIN action offer 2 sets of defaults audio prompts: - *usr/lib/sems/audio/webconference* (English) - *usr/lib/sems/audio/webconference/de* (German)

Audio file	Context	Content
setPin_welcome	Use for welcome	'welcome, you can set a pin for your personal conference room with the number' ...
setPin_welcome_set	Used to welcome when the security PIN is already set.	'welcome, your personal conference room with the number' ...
setPin_enter_pin	Used to prompt the security PIN	'please enter the security pin of the room number' ...
setPin_change_pin	Used to prompt the security PIN	'please hang up if you want to keep it, otherwise'
setPin_repeat_pin	**Used to confirm the** security PIN user	'please repeat the pin and and press the pound key'
setPin_pin_set	Used in case of success	'your pin was successfully set, thanks you.'
setPin_pin_dont _match	Used when user PIN don't match	'the pin numbers you've enter does not match. Please try again, and enter a new PIN, followed by the pound key.'
setPin_failed	Used in case of failure	'please hang up if you want to keep it, otherwise'

Two-Factor authentication ~~~~~~~~~~~~~~~~~~~~~~~~~-

Audio file	Context	Content
2fa_greeting	Use for welcome	'Please enter the two factor authentication PIN number that was set for this line
2fa_pin_correct	Used in case of success	'that is correct, thanks you. Please hold the line to be connected'
2fa_failed	Used to prompt the security PIN	'I'm sorry you're having entering the pin number. Please hold the line to be connected to the help desk.'
2fa_pin_wrong	Used to prompt the security PIN	'sorry this is not correct. Please ente rthe 2 factor authentication pin number that we set for this line.'

Chapter 12

Reference of Global Configuration Parameters

This reference lists all global configuration parameters used in ABC SBC. Note that they have default values which are designated to accommodate most use-cases and can have massive impact on operation if changed: modify them only after careful consideration. The GUI screen is showing recommended default values. When the actual value is changed, the default value is highlighted as bold text.

Important: When the global configuration parameters are updated, a warning message with a link to activate the new SBC configuration is shown in the GUI. No changes are applied until the "activate" link is used.

When the configuration changes are applied, appropriate services might be restarted (e.g. SIP and RTP processes) depending on what parameters were changed. Note that this may cause service disruption.

The configuration parameters are grouped as follows:

- *AWS Parameters*
- *Backup Parameters*
- *CDR Parameters*
- *Event Parameters*
- *Eventbeat Parameters*
- *Firewall Parameters*
- *LDAP Parameters*
- *Lawful Interception Parameters*
- *Login*
- *Low-level Parameters*
- *Miscellaneous Parameters*
- *System Monitoring Parameters*
- *PCAP Parameters*
- *SEMS Parameters*
- *SIP Parameters*
- *SRTP Parameters*
- *Syslog Parameters*
- *Conference Parameters*

- *Signaling SSL*

12.1 AWS Parameters

These parameters are used when ABC SBC is deployed on Amazon AWS. They take effect only if the ABC SBC part supporting AWS was installed, which can be enabled using "–withaws" command line option of the "sbc-install" command when installing ABC SBC.

At this moment they are used for setting up SNS-based reputation list subscription, see *AWS: Reputation Lists*.

Note that anyone in possession of an AWS IAM User Access key may impersonate the key's owner. It therefore makes sense to create a user with limited permissions and access AWS from the ABC SBC under this user's identity. Read the following link to learn more about IAM user identities: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html

Table 1: AWS Parameters

Parameter Name	**Description**
Region for AWS requests	AWS Region to which the SNS subscription requests are sent; unless needed otherwise, use the same region as in the SNS ARNs you are subscribing to. Other AWS queries than SNS are using the SBC's region, except for CloudWatch when redirected to a separate region in Userdata.
AWS access KEY ID	Key ID of an AWS user who was permission for the AWS services
AWS secret access KEY	The secret associated with the AWS user's key id. Note that the secret is only revealed when they key is created. When forgotten, the key must be created newly. When leaked, anyone in posession of the key may impersonate the user.
Perform Amazon setup steps at boot time:	This option must be enabled when the ABC SBC is operated in Amazon Elastic Cloud. When set, the instance acquires important configuration information such as its public IP address, SSH key, IAM role and more during the boot time. Note: the frafos-sbc-awinit rpm has to be installed to support this.

12.2 Backup Parameters

These parameters set ABC SBC daily backups. See also more in *Backup and Restore Operations*.

Table 2: Backup Parameters

Parameter Name	Description
Create daily Sbc configuration backups	If enabled, daily snapshot of ABC SBC configuration will be created into backup gzipped tarball file.
Include provisioned tables in daily backups	If enabled, the daily backup will include also content of whole provisoned tables.
Number of days to keep backups	Sets the retention period for backup files. All files named sbc-backup-* in the backup directory older than specified number of days will be deleted on every daily backup run. Use 0 to disable automatic deletion of old backup files.
Destination directory for backups	Specifies the destination directory for the daily backup files. Default is "/data/backups" directory.
Include sensitive system data (ssh keys,) in backup	If enabled, the system ssh daemon configuration, system ssh keys and root user authorized keys will be included in the daily backup. Disabled by default.
Run daily backups on config master node only	If enabled, the daily backups will be performed only on node which is set to be the configuration master node. Disabled by default.
Full path to extra files to include in backup	Extra custom files to be included in backup can be listed using full paths to the files, more fields separated by comma. The filename must not contain comma character.

12.3 CDR Parameters

These parameters allow to define how and where CDRs are stored. See also more in *Call Data Records (CDRs)*.

Table 3: CDR Parameters

Parameter Name	Description
Number of CDR files to keep	CDR Retention policy. The ABC SBC produces CDRs for all completed calls in CSV form. Sets number of CDR files to keep. Set to 0 to disable CDRs generation.
Directory for exported CDR files:	Directory in filesystem where the CSV CDRs are stored.
CDR files rotation frequency (daily,weekly, monthly)	Sets the frequency of CDR files rotation. Use "daily", "weekly" or "monthly". The number of rotated files to keep before deletion is set using the "Number of CDR files to keep"
Enable new version of CDRs (CDR-NG)	Enables new version of CDRs, called CDR-NG. This feature is in experimantal state in ABC SBC 4.5 release.

12.4 Event Parameters

These parameters allow to define how and where events are stored. See also more in *Events (optional)*.

Table 4: Event Parameters

Parameter Name	Description
Number of days to keep old traffic log files	Local retention policy. Particularly useful when no ABC Monitor is attached to the ABC SBC. Must be shorter than the retention policy at ABC Monitor – otherwise the ABC SBC may keep copying files that already expired at ABC Monitor. See Section *ABC Monitor Initial Configuration*

continues on next page

Table 4 – continued from previous page

ABC Monitor address	IP address or DNS name of ABC Monitor. Empty if no ABC Monitor is attached to the ABC SBC.
Secondary ABC Monitor address	IP address or DNS name of secondary ABC Monitor. Empty if no secondary ABC Monitor is attached to the ABC SBC.
Replicate traffic logs to ABC Monitor	Allows to push collected PCAPs (see Section *Diagnostics Dashboard*) to a Monitor server using the rsync protocol.
Replicate traffic logs to secondary ABC Monitor	Allows to push collected PCAPs (see Section *Diagnostics Dashboard*) to a Monitor server using the rsync protocol.
Replicate recordings to ABC Monitor	Allows to push recorded audio files (see Section *Audio Recording*) to a Monitor server using the rsync protocol.
Replicate recordings to secondary ABC Monitor	Allows to push recorded audio files (see Section *Audio Recording*) to a Monitor server using the rsync protocol.
Delete traffic logs and recordings from SBC after replicating to ABC Monitor	If enabled, the traffic log and recording files will be deleted on SBC side after replicating to the ABC Monitor server. This also reduces the network traffic from ABC SBC to ABC Monitor. The recommended configuration setting is **enabled**.
Replication rsync password	rsync password to be used for replicating traffic logs and recorded audio.
Replication rsync password for secondary ABC Monitor	rsync password to be used for replicating traffic logs and recorded audio.
Use secure TLS connection to ABC Monitor	If enabled, events, traffic log and recording files will be pushed to ABC Monitor over TLS secured connection. It is highly recommended to install trusted certificate for this on ABC Monitor end instead of default self-signed.
Verify ABC Monitor TLS certificate	Sets if remote certificate for TLS connection to ABC Monitor should be verified, if secure connection is enabled.
Number of hours to keep old recordings (0 to not delete)	Retention policy for recored WAV files.
Generate an event if a SIP transaction reaches the defined number of retransmissions	Allows to monitor failing incoming transactions and detect SIP UACs with connectivity issues. The events are of type "notice" and appear in ABC Monitor's Transport Dashboard. Use with care, a too low number will result in dramatic increase of events. If used, recommended value is 4.
Maximum number of events buffered in local Redis	Retention policy for locally buffered events
List of call variables added into events	Contain list of call variables that are added into call events. See *Call Processing Events*. The list shall contain comma separated pairs: *<var_name>:<flag>* where *<var_name>* is name of call variable and *<flag>* is 0 or 1 specifing whether the value of call variable can be overwritten. User may use the wildcard (*) charactere to denotate ALL events.
Generate an event on UDP receive buffer errors	If enabled, alert event will be generated if UDP receive buffer errors are detected on system network interface.

continues on next page

Table 4 – continued from previous page

Generate an event on UDP send buffer errors	If enabled, alert event will be generated if UDP send buffer errors are detected on system network interface.
Generate an event on UDP packet receive errors	If enabled, alert event will be generated if UDP packet receive errors are detected on system network interface.
Generate an event on IP incoming packet receive errors	If enabled, alert event will be generated if IP incoming packet receive errors are detected on system network interface.
Generate an event on outgoing packets dropped errors	If enabled, alert event will be generated if outgoing packets dropped errors are detected on system network interface.
Alarm when number of calls reaches % of the license.	sems will yield a warning message once the number of session reached X% of the license limit. A downstream message is also yield (info level), once the number of session go below X%. Default: 75.

12.5 Eventbeat Parameters

These parameters allow to tweaks and debug the event communications between an ABC SBC node and an ABC Monitor one. Some statistics may be generated and exposed on the application interface TCP port (*:4245* and *:4246*).

Table 5: Eventbeat Parameters

Parameter Name	**Description**
Event batching size	Maximum number of event sent at once to the monitor.
Enable enventbeat statistic reporting	Expose on the tcp port (*:4245* for *sbc-eventbeat-1* and *:4246* for *sbc-eventbeat-2* some live metrics about events processing.
Interval between each statistic	Interval on which a single statistic entity was recorded.
How many statistic entries per payload	How many statistics entities should be returned in a single payload. Ex: To have statistic about the last minutes, per packets of 5 seconds, set the following : • *Interval between each stat* to 5 • set *How many entries per payload* to 12

12.6 Firewall Parameters

Table 6: Firewall Parameters

Parameter Name	**Description**
Drop UDP signaling packets not looking like SIP	If enabled, any UDP packets not bearing a "SIP signature" in first 200 bytes will be discarded without further notice.
Blacklist IP addr for repeated ssh login failures	If enabled, IP address of failed ssh login attempt will be put on blacklist, silently dropping all packets from it.

continues on next page

Table 6 – continued from previous page

Time in seconds to check for repeated ssh failures	Sets the time period during which the failed ssh login attempts are counted when considering adding IP address on blacklist.
Maximum number of ssh failures before blacklisting	Sets the number of failed ssh login attempts, after which the IP address is added to blacklist.
Time in seconds to blacklist IP addr for ssh failures	Sets the time how long the IP address will be held on blacklist, before removing it from blacklist automatically.
Blacklist IP addr for repeated gui login failures	If enabled, IP address of failed Sbc gui login will be put on blacklist, silently dropping all packets from it.
Time in seconds to check for repeated gui login failures	Sets the time period during which the failed gui login attempts are counted when considering adding IP address on blacklist.
Maximum gui login failures before blacklisting	Sets the number of failed gui login attempts, after which the IP address is added to blacklist.
Time in seconds to blacklist IP addr for gui login failures	Sets the time how long the IP address will be held on blacklist, before removing it from blacklist automatically.
Blacklist IP addr for repeated signaling failures	If enabled, IP address of request that failed authentication, exceeded limit, failed sanity check, was dropped by Drop action or Log message for replies action was used, will be put on blacklist, silently dropping all packets from it. Note that the individual reasons for blacklisting have to be also enabled in CA settings or in the Drop or Log message for replies actions parameter. See Section *Automatic IP Address Blocking* for more details.
Signaling failures blacklist: IP address start score before any offense	Sets the score used as a starting value before any offense has been registered. This start value will be decreased each time until it reaches 0 or less, which finally leads to the blacklisting of the incriminated IP address. See Section *Automatic IP Address Blocking* for more details.
Signaling failures blacklist: rate per second used to calculate a time-related bonus between offenses	Sets the allowed rate of offenses in events per second. This allows the score to recover slightly over time and thus can be understood as a bonus for good behavior. See Section *Automatic IP Address Blocking* for more details.
Signaling failures blacklist: time in seconds to remove entries for which no event has occured from score calculation	Sets the number of seconds after which, if no offense from a certain IP address has been seen, that IP address is removed from the scoring table. Should a new offense be registered from a deleted IP address, the start score will be used. This allows for keeping the scoring table at a reasonable size. See Section *Automatic IP Address Blocking* for more details.
Time in seconds to blacklist IP addr for signaling failures	Sets the time how long the IP address will be held on blacklist, before removing it from blacklist automatically (for drop, failed auth, limit, sanity). See Section *Automatic IP Address Blocking* for more details.
Greylist: time delay in seconds to give IP a chance to prove validity	If the traffic from IP address proves validity during this probation period, the source IP addr will be added to whitelist. Note that the corresponding action options like “Greylist IP address” or “Log to greylist” have to be used. See Section *Automatic Proactive Blocking: Greylisting* for more details.

continues on next page

Table 6 – continued from previous page

Greylist: time period in seconds when IP can be blacklisted if repeats and did not prove validity	If traffic from IP address did not prove validity during the probation time period, and new packet comes during this time period since first packet, the source IP addr will be added to blacklist. Note that the "Greylist" flag has to be enabled on ABC SBC signaling interface for this to work. All traffic from the IP addresses on blacklist will be silently dropped. See Section *Automatic Proactive Blocking: Greylisting* for more details.
Greylist: time in seconds to keep IP on blacklist	Sets how long to keep the IP address on blacklist. After this time it is removed from blacklist and has a chance to prove validity again. See Section *Automatic Proactive Blocking: Greylisting* for more details.
Greylist: time in seconds to keep IP on whitelist	Sets how long to keep IP address on whitelist. After this time it is removed from whitelist and has to prove validity again. See Section *Automatic Proactive Blocking: Greylisting* for more details.
Greylist: additional ports or port ranges (a:b) to check in addition to signaling ports, space separated	Sets additional ports to ports defined on ABC SBC signaling interfaces. If used, traffic coming to this port(s) will be also subject to the greylisting procedure. You can specify single port(s) or port ranges (in format lower:higher), space separated. See Section *Automatic Proactive Blocking: Greylisting* for more details.
Blacklist: Log blacklisted IP addresses to syslog	Log blacklisted IP addresses to syslog. Entries are logged in the following file: '/var/log/frafos/sems-blacklist.log'
Greylist: Log greylisted IP addresses to syslog	Log greylisted IP addresses to syslog Entries are logged in the following file: '/var/log/frafos/sems-greylist.log'
Overall limit in packets per second from not approved IP addresses	This option can be used to set overall packets per second limit on all IP addresses, that did not prove validity using "Greylist IP address" or "Log to greylist" action options. Use with caution. Use 0 to disable any rate limiting.

12.7 LDAP Parameters

This is configuration of LDAP server for LDAP based authentication.

Table 7: LDAP Parameters

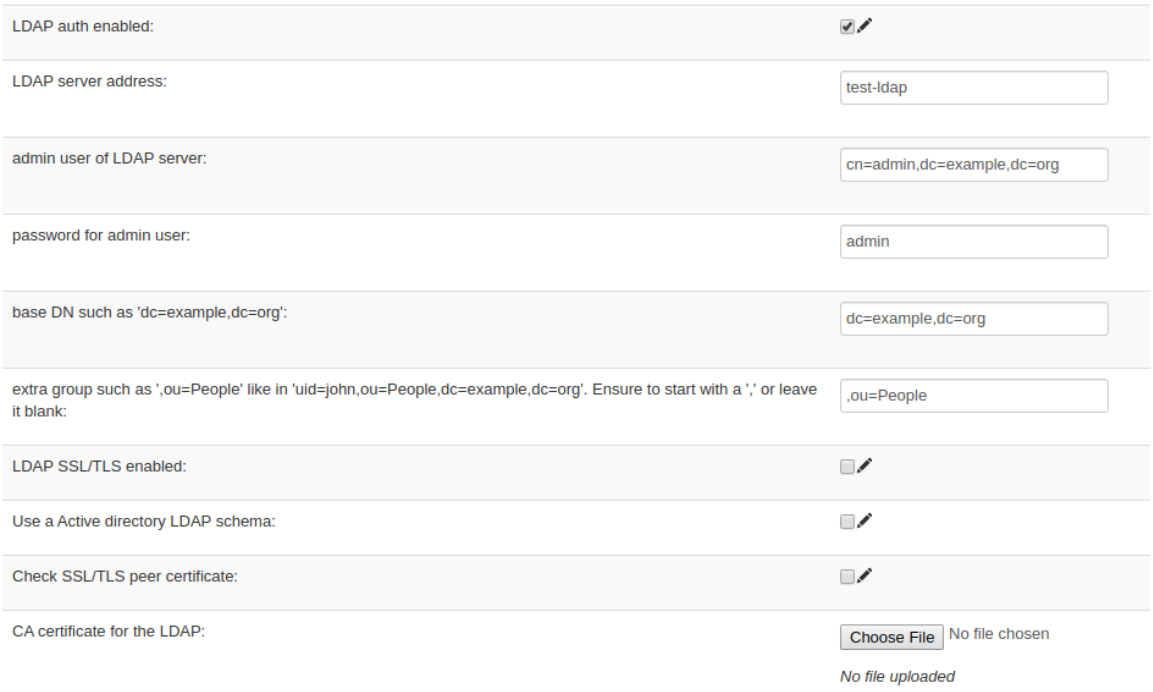
Parameter Name	**Description**
LDAP auth enabled	Enable LDAP authentication
LDAP server address	LDAP host on which the LDAP service can be reached (ldap://IP:PORT or ldap://IP or ldap://my.domain)
LDAP distinguished name / admin user DN	Specifies the distinguished name with which to bind to the directory server for lookups
LDAP credentials / admin user PW	Specifies the credentials with which to bind. Should work on login with the LDAP distinguished name
base DN such as 'dc=example,dc=org'	Default search DN of the LDAP. In "cn=admin,dc=example,dc=org" the default DN is "dc=example,dc=org"

continues on next page

Table 7 – continued from previous page

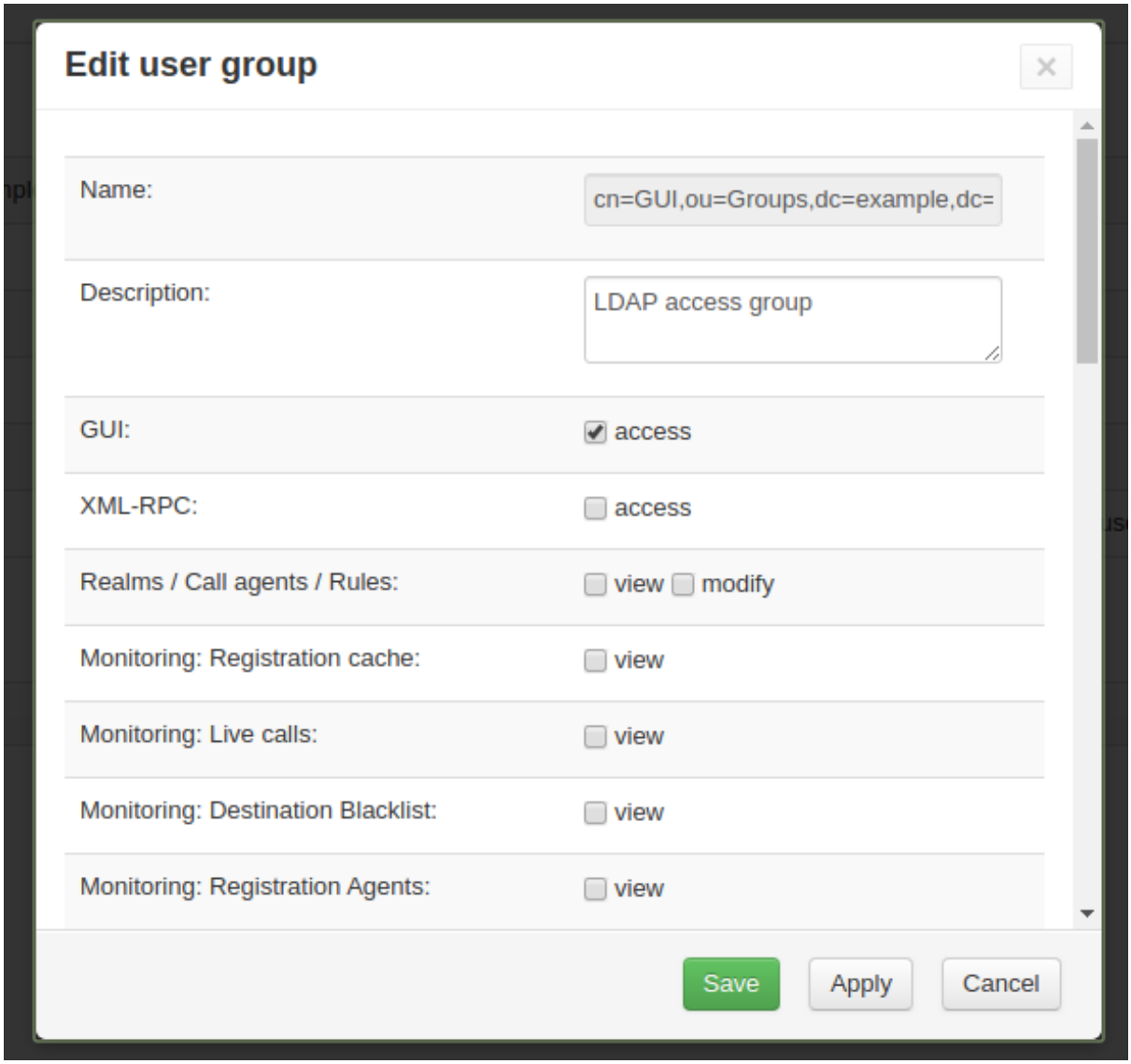
extra group such as 'ou=People' like in "uid=john,ou=People, dc=example,dc=org"	For users dn such as "uid=john,ou=People,dc=example,dc=org", the value of this var should be "ou=People" so the users only provide their uid ("john" in this case) & password to log. Note that to complete a user login, the ldap user must be a memberof a group named after a GUI group that support login. This group must be a primary group of that user.
SSL/TLS enabled	Enable the SSL/LDAP connection. Some LDAP may required that you manually place the PEM certificate into the */etc/openldap/certs* directory
Active Directory LDAP	Use an Active Directory LDAP server
Check SSL/TLS peer certificate	Enable the check of client certificates. Please note that an Active Directory LDAP **needs** the certs to be configured in */etc/openldap/certs*
CA certificate for the LDAP	List of certificates to which the client's one are check. The certificate must be in PEM format.

Following, a simple ldap setup :



There is a docker container available on github that match the screenshot configuration : https://github.com/burgesQ/simple-ldap.

Note that to support the login of the *John Doe* user, you'll need to create a GUI group named after the full dn of the LDAP group (cn=GUI,ou=Groups,dc=example,dc=org).



You can then login with the credential *john* and the password *johnldap*

12.8 Lawful Interception Parameters

This is configuration of Lawful Interception.

Table 8: Lawful Interception Parameters

Parameter Name	Description
Lawful Interception enabled	Enable the feature generally. Note that it has to be used also under corresponding action to take effect.
Operator ID	Set the Operator ID value.
Delivery Country Code (DCC)	Set the Delivery Country Code (DCC) value.

12.9 Login

Parameters related to login/logout.

Table 9: Login Parameters

Parameter Name	Description
GUI auto-logout time	Timeout in minutes of inactivity after which the GUI user is automatically logged out. Use '0' to disable auto-logout
Max failed login	Maximum number of failed logins till the user account is blocked. This is for brute force hacking protection. Use '0' to disable account blocking due to failed logins.
Blocking period	How long the user account is blocked (in seconds) if number of invalid logins reach the 'Max failed login'
Garbage collect timeout	Timeout (in days) after which the data used for brute force hacking protection are removed from DB.
Time for terminal session automatic logout if idle, in seconds	Sets the time in seconds after which idle terminal session to ABC SBC will be automatically closed. Default value is 600 sec. Use 0 to disable.

12.10 Low-level Parameters

These settings have effect only after reboot of the server. Additional information can be found in the Section *Hardware Specific Configurations*.

Caution: changing these parameters may dramatically change system behaviour. Their effect largely depends on used equipment.

Table 10: Low-level Parameters

Parameter Name	Description
Interfaces where to enable RPS	Network interfaces on which a "receive packet steering" kernel feature should be enabled, separated by spaces. While the kernel leaves this option by default off, turning it on can increase media throughput.
Interfaces where to set ethtool options	Network interfaces where to apply the following coalesce and ringbuffer ethtool options. Separated by spaces.
Coalesce ethtool options	Ethernet adapter coalescing options, syntax of ethtool. Applied on interfaces listed in "Interfaces where to set ethtool options". This option allows to fine-tune a trade-off between less-CPU-intensive and more-real-time packet processing in kernel. The tuning outcome is specific to used network card.
Ringbuffer ethtool options	Ethernet adapter rx/tx ring parameters, syntax of ethtool. Applied on interfaces listed in "Interfaces where to set ethtool options". Fine-tuning this parameter is specific to used network card. Increasing buffer sizes allows to deal with temporary packet bursts, while latency may increase.
Interfaces where to bind irqs to CPUs	Network interfaces on which the individual interrupts for receive and transmit queues should be statically bound to individual CPUs / CPU cores. This option may increase media throughput on network cards with multiple queues.

continues on next page

Table 10 – continued from previous page

Run db check on boot	If enabled, run "mysqlcheck" command during boot process. This option allows a safe recovery from an unexpected shutdown and is therefore by default turned on. The check may slowdown machine startup.
Clean tmp files on boot	If enabled, clean-up system directory for temporary files.
Sems memory limit in % from total memory	Limit Sems process memory maximum usage. Set to 0 for no limit.
Writeback after dirty bytes	Max. ammount of data that system can cache before writing all back disk is initiated. Set to 0 to not update. Large cache can improve system performance during heavy writes. At the same time, risk of data loss in the case of unexpected failure is higher. The performance trade-offs may be particularly important on virtual machines with heavy writes, like when PCAP or WAV files are recorded. Various linux fora[1][2] discuss this kernel option "vm.dirty_bytes".
Background writeback after dirty bytes	Amount of data system can cache before background writing back to disk is initiated. Set to 0 to not update. Like with the previous option, increased buffering provides better performance in load bursts and increased risk of data loss. Various linux fora[1][2] discuss this kernel option "vm.dirty_background_bytes".
Provisioned tables redis disk persistence time interval (in seconds)	Sets the time interval after which provisoned tables data on Sbc slave node will be saved from in-memory redis database to disk to allow persistence for reboot. May be tuned according to provisioned tables data size. The data is saved if there were more record changes than set via the following setting for minimum number of records. Default is 600 seconds.
Provisioned tables redis disk persistence number of records to trigger save	Set minimum number of provisoned tables record changes that trigger save to disk. The data will be saved when both the number of changed records and the time interval conditions are met. Default is 1 record.
Use real-time priority on provisioned tables redis	If enabled, will use real-time process priority on provisioned tables redis db, which helps performance. Can be used only if operating system or container permissions support this.

[1] See reports on linux write caching https://lonesysadmin.net/2013/12/22/better-linux-disk-caching-performance-vm-dirty_ratio/
[2] See Linux virtual memory subsystem options: https://www.kernel.org/doc/Documentation/sysctl/vm.txt

12.11 Miscellaneous Parameters

Table 11: Miscellaneous Parameters

Parameter Name	Description
Permit root login using ssh	Sets if root is allowed to login to ABC SBC server using ssh. Use 'yes' to allow root login, or 'prohibit-password' to allow login but password and keyboard-interactive authentication disabled, or 'no' to disable.
Blacklist timeout for IP addresses from external sources	Timeout in seconds for the IP addresses blaclisted by RESTFul requests.
Enable sending important syslog entries to ABC monitor	Enables or disables sending syslog entries of levels 'critical' up to 'emergency' as an alert to the ABC monitor.
Firewall control using grpc on container	Enables firewall control using grpc protocol, takes effect only when running on container. Disabled by default.
Automatically add new nodes	If enabled, records for new nodes that pull config from configuration master will be automatically added. If disabled, the configuration master will refuse to provide configuration to nodes that are not already defined in Nodes configuration.
Session Management enable	Enables advanced load-balancing, see more details in the section :ref:Sec-adv-load-balancing
Failed system login lock unlock time	Time in seconds to keep system accouns locked after 3 failed login attempts. Default value is 600 seconds.
Geoip - license file for geoipupdate command	Used to pass license file for geoipupdate command, which is run periodically if the license is provided to retrieve geoip GeoLite 2 database. The license has to be created by user using his MaxMind account. When creating the license, select version usable with geoipupdate version 2.5.

12.12 System Monitoring Parameters

These parameters allow to set up an email alarm if system resources are used excessively.

Not that this same email is used to setup the Let's encrypt auto certification.

Table 12: System Monitoring Parameters

Parameter Name	Description
email for sending alerts	Email address to which important alerts like reports on excessive CPU usage are sent. Use empty value to disable sending the email alerts. This email address will be also used in case of let's encrypt auto certificate renew on TLS profile.
mailserver for sending alerts	By default the email is passed to a local SMTP relay. Note: when ABC SBC is running in container, mail reley on localhost is not available and external mail server has to be used.
from address for sending alerts	email address used for From in email alerts, system default is used if empty
1min load threshold	CPU load threshold which if exceeded for one minute will raise an alarm. The load threshold values should be set correspondingly to system CPU cores number.
5min load threshold	CPU load threshold which if exceeded for five minutes will raise an alarm (typically lower value than previous). The load threshold values should be set correspondingly to system CPU cores number.
cpu wait % threshold	threshold for % of CPU time in wait status to raise an alarm
memory usage % threshold	threshold of memory occupation in % which if exceeded will raise an alarm
disk usage % threshold	threshold of disk usage in % which if exceeded will raise an an alarm
send system monitoring data to ABC Monitor	if remote ABC monitor is used, send system monitoring data to it together with signaling events
send extended system info emails when over treshold	if enabled, email with more detailed system information will be sent when some monitoring threshold is reached
extended info emails frequency	limit frequency of sending the extended info emails, use value with min, hour or day suffix
Check status of system interfaces	If enabled, system network iterfaces will be periodically checked and alert events created if errors are detected. Individual check types can be set using following options.
Include only MI/SI interfaces in status check	Sets if the system network interfaces check includes only media and signalling interfaces, or all.
Interval for the system interfaces cheks	Sets the frequency of the system network interfaces checks, in seconds. Default value is 300 sec.

12.13 System Monitoring LDAP access Parameters

The ABC Monitor may use an LDAP server to authenticate user. To configure it, access ABC Monitor GUI Settings / General page and set the LDAP settings under "Authentication" section.

12.14 PCAP Parameters

These parameters allow to set up how the most recent SIP traffic is recorded on the system for sake of troubleshooting. The ABC SBC stores the SIP traffic in PCAP files of given size and deletes the least recent files. The PCAP files can be inspected in the administrative interface as shown in Secion *User Recent Traffic*.

Table 13: PCAP Parameters

Parameter Name	Description
File size in MB for one pcap file	maximum size of a PCAP file after which a new file is created
Number of pcap files to keep	PCAP retention policy. PCAP files are rotated and only the configured numer of PCAP files is kept. The least recent files are deleted. Use 0 to disable storing SIP traffic completely, which is not recommended because of troubleshooting. Note: the pcap filenames are using extension ".pcapXX" where XX corresponds to the file number. If the number of files is lowered, the older files which are not going to be overwritten by next cycle are NOT deleted, and have to be deleted by admin.

12.15 SEMS Parameters

These parameters determine the behaviour of the ABS-SBC "engine", the SEMS signaling and media processor. The parameters are used primarily for troubleshooting and performance tuning and shall be therefore changed only when there is a good reason for doing so.

Table 14: SEMS Parameters

Parameter Name	Description
Session processor threads	These threads process the SIP signaling of the sessions. They also process the B (routing) and C (outbound) rules of the ABC rule set. created in a thread pool among which all SIP sessions are distributed. Usually we recommend to set this to the number of usable hardware threads on the CPU multiplied by two, but to no less than 8 threads. If the SBC needs to process a lot of external data in the routing or C rules, e.g. needs to query provisioned tables or external API server via REST, then it is recommended to set this to a high number.
Media processor threads	These threads process RTP media for transcoding and media applications like conferencing and announcements. In normal SBC operation, when those functionalities are not used, these threads will be idle. Like the session processor threads, the number configured here sets the number of threads created in a thread pool among which all media sessions are distributed. If transcoding or media applications are used, it is recommended to set this number to two times the usable CPU hardware threads, otherwise it is recommended to leave them to the default (16) or even less.

continues on next page

Table 14 – continued from previous page

SIP server threads	These threads receive SIP messages from the network and initially parse them for later processing by the Session processor threads, immediately reply e.g. if the reply is given by the SIP dialog state (e.g. errors). They also process the A rules of the ABC rule sets. The number of threads configured here is started for every signaling interface (SI), and one set for udp and one for tcp; so e.g. if five SI interfaces are configured, and this is set to 4, then 5*4*2=40 threads are started. The recommended number depends on the number of signaling interfaces; e.g. on a setup with two signaling interfaces, the recommended number would be equal to the number of CPU cores (e.g. 8, 16 or 32). On a setup with many signaling interfaces, this should be set to e.g. 2 or 4.
RTP receiver threads	These threads receive RTP packets and relay them. They also decrypt SRTP packets if enabled. As with the thread pools above, this number is a global number of threads for a thread pool. The recommended number to set this to is two to four times the usable CPU hardware threads.
Call restore threads (HA)	This is a thread pool that is only used when doing the call restore after failover. It is recommended to set it to the number of usable CPU hardware threads.
Out-of-dialog requests threads	These threads handle REGISTER, SUBSCRIBE/NOTIFY and MESSAGE requests. If a lot of registrations are handled, or a lot of subscriptions, then it is recommended to set this to a higher number.
Force symmetric RTP for mediaserver apps:	If enabled, embedded media processing actions will ignore IP addresses in callers' SDP and send its RTP to wheere caller's RTP came from.
RTP keep-alive frequency	Defines how often if at all ABC SBC sends RTP keepalive packets to its peers. See *Setting RTP Inactivity Timer and Keepalive Timer*.
RTP timeout	Defines period of time after which a call is terminated if RTP packets stop arriving. See *Setting RTP Inactivity Timer and Keepalive Timer*.
Use raw sockets	Performance optimization techniques for sending RTP packets on linux systems with slow UDP stack.
Default Destination Blacklist TTL	Defines how long are unavailable IP destinations maintained on a blacklist to which no SIP traffic is sent by default. For Call Agent, a specific value may be entered in the Call Agent parameters. See *IP Blacklisting: Adaptive Availability Management*.
Persistent redis storage	If enabled, the calls state data that is stored in redis db, will be preserved during server reboot.
Load q850_reason call control module	If enabled, the module for processing Q.850 reasons will be loaded. It can be used only if custom cc_q850_reason.conf file is provided.

continues on next page

Table 14 – continued from previous page

Terminate dialog upon failure replies for in-dialog OPTIONS	Terminate dialog if in-dialog OPTIONS request fails with reply that should cause dialog termination. Reply codes that should terminate the dialog according to RFC 5057 are: 404, 410, 416, 482, 483, 484, 485, 502, 604. Additionally ABC SBC handles following replies the same way as those listed above: 408, 480. Affects only INVITE based dialogs (i.e. calls). The purpose of this option is to cope with interoperability issues caused by badly implemented SIP user agents that can't handle in-dialog OPTIONS correctly. Default value: on (terminate the dialog)
Remove filtered m-lines	Remove media lines filtered out by media whitelist/blacklist. These lines are left in SDP but marked as inactive if not enabled. This option is applied globally on all calls with active media whitelist or blacklist (see *Media Type Filtering*). The purpose of this option is to cope with interoperability issues caused by badly implemented SIP user agents that can't handle inactive media streams correctly. Default value: off (i.e. mark media lines as inactive)
Filter forced transports	Remove media lines that do not match outbound transport forced by Force RTP/SRTP action (see *RTP and SRTP Interworking*). These lines are left in SDP but converted to the required trasport if not enabled. For example: Caller is sending one audio stream over RTP and another audio stream over SRTP (comonly used when SRTP is configured as optional on a phone). SRTP is forced in outbound rules on ABC SBC. If Filter forced transports option is "off" ABC SBC forwards SDP with two audio streams to the callee both of them over SRTP. If this option is "on" ABC SBC forwards SDP with just one audio stream over SRTP to the callee. This option is applied globally on all calls using Force RTP/SRTP action. The purpose of this option is to cope with interoperability issues caused by user agents that can't handle multiple media streams of the same type. Default value: off (i.e. convert the media lines to the forced transport)
Call transfers using late offer-answer	Use offer-less INVITE when generating new call leg during call transfer (unattended call transfer or call transfer replacing non-local call). It is probably the only reliable way that should work. Unfortunatelly too many SIP UAs do not implement late offer-answer correctly. Default value: off

continues on next page

Table 14 – continued from previous page

Predefined payloads for call transfers	Coma separated list of codecs to be added into SDP of INVITE generated during call transfer (unattended call transfer or call transfer replacing non-local call). If no codecs are listed, only codecs used within the call are used what can cause troubles if the destination doesn't support these. Only simple codecs can be used (no parameters can be specified). For example: pcmu,pcma Default value: empty
Learn remote media address interval	Interval (in milliseconds) after first RTP packet received in which RTP address may still change and will be re-learned. I.e. after that interval SEMS locks on the remote address. Especially for re-learning after re-Invite, this may prevent locking on the old address due to some late RTP packets from the old remote address. Default value: 0 ms (disabled), lock on the first packet
Mariadb timeout for "Read call variables" queries	Timeout (in seconds) of Mariadb queries done when reading call variables using the action or condition "Read call variables". The main purpose of this parameter is to reduce problems caused by queries that may take too much time and block processing of other calls. Please note that timeout of such Mariadb queries means that system is either overloaded or blocked and the root cause should be fixed instead of tuning the timeout value. Negative value or 0 means that default timeout of the MySQL++ library will be used. Default value: 5
Log level	This option changes the SEMS syslog globally. See the Section *Reference of Log Level Parameters* for a full list of options.
Force outbound interface	If enabled, UDP packets sent will be forced to use the system interface attached to the outbound call agent. Please note that this option relies on operating system capabilities that have heavy limitations. Especially, when forcing the outbound interface, the Linux IP stack will set the source IP on its own, which might lead to unwanted effects (invalid source IP that e.g. SEMS might not be using at all). In many cases, this option will not effect the desired functionality and is not recommended. Manually configured source IP based policy routing is the prefered method. Default value: off
Dump TLS session keys to file	If enabled, the TLS session keys will be dumped to a file for diagnostics (into directory /data/pcap/tls_keys). Disabled by default. Requirements: note that this option must be enabled if one wishes to download from the GUI a bundle composed of pcap files and tls keys. Otherwise, the bundle may only contain pcap files. Limitations: WebRTC interface isn't supported.

continues on next page

Table 14 – continued from previous page

Recording playout buffer type	Type of playout buffer used for data synchronization while recording into a WAV file. Possible values: • adaptive Sophisticated playout buffer that should be more appropriate from user's perspective, especially with higher jitter and packet loss showing in the RTP stream. • simple Basic buffering that might not be sufficient with lossy line. Default value: adaptive
SIPREC outbound interface	Use Sbc interface name to force outbound interface for SIP recording. Leave empty by default.
SIPREC media interface	Sbc interface to be used for sending media towards SIPREC server. Empty by default.
SIPREC SIP timer A (ms)	SIP timer A used towards SIPREC server. Default value 500ms.
SIPREC SIP timer B (ms)	SIP timer B used towards SIPREC server. Default value 32s.
SIPREC SIP timer C (ms)	SIP timer C used towards SIPREC server. Default value 180s.
SIPREC SIP timer F (ms)	SIP timer F used towards SIPREC server. Default value 32s.
SIPREC SIP timer L (ms)	Timer L used towards SIPREC server. Default value 32s.
SIPREC SIP timer M (ms)	Timer M used towards SIPREC server. Default value 8s.
SIPREC SIP timer T2 (ms)	SIP timer T2 used towards SIPREC server. Default value 4s.
Minimal supported TLS version	The minimal supported TLS version on signaling interfaces. Use tls1 or tls1.1 or tls1.2.
TLS cipher list	The supported TLS ciphers list for signaling interfaces, in openssl syntax.
TLS EC curves list	Allows for setting the EC curves used with TLS for signaling interface. The string is a colon separated list of curve NIDs or names, for example "P-521:P-384:P-256".
Websocket ping-pong interval in seconds	Interval in seconds to send keepalive ping-pong messages on websocket signaling interfaces. Use 0 to disable.
Soft limit for out-of-dialog transactions	Number of active server transactions that, if passed, will trigger an alert event. This limit will only be taken into consideration when creating a server transaction which is not related in any way to an existing dialog. Use 0 to disable that feature. See section *Server Transaction limits* for more details.
Hard limit for out-of-dialog transactions	Limit for the number of active server transactions, which will be enforced when creating a new server transaction not related to an existing dialog. The limit is enforced by replying to new requests with "503 Overloaded". Additionnally, a corresponding monitoring event will be created. Use 0 to disable that feature. See section *Server Transaction limits* for more details.

continues on next page

Table 14 – continued from previous page

Event throttling for soft/hard OOD limit	Throttle the events generated by the hard & soft limit for out-of-dialog transactions to no more than one of each type (soft / hard) per configured time lapse in seconds. Use 0 to disable that feature. See section *Server Transaction limits* for more details.
Soft limit for in-dialog transactions	Number of active server transactions that, if passed, will trigger an alert event. This limit will only be taken into consideration when creating a server transaction related to an existing dialog. Use 0 to disable that feature. See section *Server Transaction limits* for more details.
Hard limit for in-dialog transactions	Limit for the number of active server transactions, which will be enforced when creating a new server transaction related to an existing dialog. The limit is enforced by replying to new requests with "503 Overloaded". Additionnally, a corresponding monitoring event will be created. Use 0 to disable that feature. See section *Server Transaction limits* for more details.
Event throttling for soft/hard DLG limit	Throttle the events generated by the hard & soft limit for in-dialog transactions to no more than one of each type (soft / hard) per configured time lapse in seconds. Use 0 to disable that feature. See section *Server Transaction limits* for more details.
Loop detection secret	This parameter is used to create a special branch tag. When we receive a new request that contains our prepared tag in the first Via-HF, we refuse the request with 482 Loop detected. Use empty value to disable, "auto" for automatic secret string, or provide a string. Deafult is "auto".
Strict checking of the user part of a URI to only allow chars as per RFC3261	When the strict checking is enabled, the user part of a URI is only allowed to contain the chars as per RFC3261 (see ABNF rules). When disabled, ABC SBC does everything to let the most through, as long as it does not prevent it from parsing URIs correctly. Enabled by default.
TCP connection idle timeout in milliseconds	Sets TCP connection timeout if idle, on signaling interfaces. Use value in milliseconds, or 0 to disable.
Delay after startup to ignore limits	Delay in seconds to ignore CAPS and other limits after start of ABC SBC signaling application.
RESTful interface - verify https peer	If enabled, the validity of https certfificate of peer will be verified on RESTful interface queries. Enabled by default.
REST Custom CA file	If provided, sems's rest module will use the provided custom CA for every outgoing https request. ABC SBC will handle on it own the adding of the ca to the nodes trusted chain. Process: • CA copyed to */etc/pki/ca-trust/source/anchors/* • run *update-ca-trust*
User-agent string	If provided, the input here is used to set the User-Agent on SIP messages.
TCP send timeout for signaling interfaces	Set TCP connection timeout in milliseconds for signaling interfaces (see TCP_USER_TIMEOUT in tcp(7) man page).

continues on next page

Table 14 – continued from previous page

Unprocessed events limit	If the REDIS server is offline, DB writes are queued internally. If REDIS is offline for a long time, the internal write queue can grow, using up a lot of memory. This parameter limits the write queue size; if the write queue has reached this size, further writes are ignored. Set to 0 to disable the limit.
Unprocessed events limit warning threshold	If the write queue grows over this threshold, SEMS warns by generating WARN level syslog messages. Set to 0 to disable.

12.16 SIP Parameters

These parameters set SIP timers, as defined in RFC 3261. All values are in ms.

An extra parameters is avaiable, see the following table:

Table 15: SIP Parameters

Parameter Name	Description
Add Q850 header to timer expiration's CANCEL.	If enable, then a Q850 header is added to the CANCEL generated by a timer expiration. Currently, only time C is supported.

12.17 SRTP Parameters

These parameters define the security handshake of Secure RTP. SRTP is always used for WebRTC and is used with some encryption-enabled SIP devices.

Table 16: SRTP Parameters

Parameter Name	Description
DTLS certificate file	Certificate file. Optional. Keep empty for self-signed certificate. That's the recommended configuration: other certificates may cause DTLS packets to become too large and consequently fail to traverse NATs due to IP fragmentation.
DTLS private key file	Private key file. Optional.
SRTP crypto-suite AES_CM_128_HMAC_SHA1	Enables / disables the corresponding crypto suite. It should be left enabled unless required otherwise for interoperability.
SRTP crypto-suite AES_CM_128_HMAC_SHA1_80	Enables / disables the corresponding crypto suite. It should be left enabled unless required otherwise for interoperability.
SRTP crypto-suite AES_256_CM_HMAC_SHA1_80 (SDES only)	Enables / disables the corresponding crypto suite. It should be left enabled unless required otherwise for interoperability.

12.18 Conference Parameters

These parameters allow to fine-tune the behaviour of SEMS webconferencing.

Table 17: Conference Parameters

Parameter Name	**Description**
Notification sender (email address)	Email address used to send notification. Default: *notificator@frafos.com*
SMTP server	SMTP server address. Format is *smtp://[username]%[domain]:[password]@[uri]:[port]* Default: *smtp://username%40domain.com:password@smtp.server.com:587/*
Enable access to conference GUI	
Enable SSL for conference websocket connection	
Ask for PIN separately from the room number	If this is not checked, the PIN is the room number, so the caller is only asked to enter one number (PIN = room number). If this is checked, there is two numbers: room number and PIN. Each room has then a PIN. If the rooms are created ad-hoc (i.e. by dialing in, without a web interface that controls the conference bridge), then the caller is first asked for the room number, and then if it doesn't exist is asked whether to create the room and enter a PIN number for that room. If the room exists then the PIN is asked and the caller is only allowed in after entering the correct PIN for that room.

12.19 Syslog Parameters

These parameters allow to fine-tune behaviour of syslog daemon. This is primarily useful when the syslogs are configured to be sent to an external system.

Table 18: Syslog Parameters

Parameter Name	Description
Syslog facility	Name of syslog facility to use for logs from the main SBC processes. Possible values are 'daemon', 'user', 'local0', 'local1' ... 'local7'.
Enable remote syslog servers	If turned on, syslog messages will be sent to an external syslog host(s) additionally to the local filesystem.
Remote syslog server address	Address of the external syslog server.
Remote syslog server port	Port number on which the external syslog server listens.
Remote syslog transport	Transport protocol on which an external syslog server listens. Use 'udp' or 'tcp'.
Log level for remote syslog server	Log messages above this level will be sent to the external syslog server. Use one of 'emergency', 'alert', 'critical', 'error', 'warning', 'notice', 'info', 'debug'.
Log files rotation frequency	Sets the interval for log files rotation. Use "daily", "weekly" or "monthly".
Number of old log files to keep	Sets the number of rotated log files to keep before deletion.
Secondary remote syslog server address	Address of the secondary external syslog server. Use empty value to not use secondary external syslog server.
Secondary remote syslog server port	Port number on which the secondary external syslog server listens.
Secondary remote syslog transport	Transport protocol on which secondary external syslog server listens. Use 'udp' or 'tcp'.
Log level for secondary remote syslog server	Log messages above this level will be sent to secondary external syslog server. Use one of 'emergency', 'alert', 'critical', 'error', 'warning', 'notice', 'info', 'debug'.
Send CDRs to remote syslog server	Enables or disables including the CDR entries in the log messages sent to the remote syslog server.

12.20 Signaling SSL

Table 19: Signaling SSL Parameters

Parameter Name	Description
Revoked certificates (CRL) file	CRL file holding a list of revoked certificates. Used by sems signaling process only.

Chapter 13

Reference of Log Level Parameters

In several ABC SBC configuration places, the log reporting levels may be configured. The ABC SBC allows to set the logging levels both globally and by functional areas. The increase log level may help with troubleshooting however caution is advised. Increased log level can dramatically degrade system performance.

This reference provides explanation how to set the proper logging level. Log levels are represented with an integer value and have the following possible values:

- 0 / ERROR
- 1 / WARNING
- 2 / INFO
- 3 / DEBUG

If only log-level is set, it is used globally. The log level can be changed however for only some specific functional area by preceding the value with "Category:Subcategory=" expression. Multiple such expressions can be combined with each other using semicolon as shown in the following example:

```
1;SIP:Transaction=3;SDP:Parser=3;RTP:*=3;PLUGIN:sbc=3
```

This example sets the default log level to 1, whereas SIP transaction machine, SDP parser, RTP engine and SBC logic reports at log level 3.

Table 1: Log Level categories

Category	Subcategory
Core	• Main • Config • Thread • Timer • Events • SessionContainer • SessionProcessor • SessionWatcher • MediaProcessor • Plugin • Utils

continues on next page

Table 1 – continued from previous page

SIP	• Ctrl • Parser • Transport • Transaction • Dialog • OfferAnswer • Session • Registration • Subscription • DNS • Blacklist
B2B	• B2BSession • B2BMedia
SDP	• Parser • MimeBody
RTP	• Stun • RtpPacket • RtcpPacket • RtpTransport • RtpStream • RtpAudio
SRTP	• SRTP • SDES • DTLS • ZRTP • Socket
AUDIO	• Audio • AudioFile • AudioMixer • Conference • Playlist • Prompt • Jitter
PLUGIN	• sbc • redis_store • websock • reg_agent • cc_gui • cc_gui_rules • sbc_replication • webconference

13.1 Debug log level per node or per system

There is an option to set a debug level for all components either per whole system or just per single node. Debug log level will be enabled or disabled for following applications: sems, xmloredis, pkapman, gocertbot, gominstrator, statman, prov2json, json2redis, webconf-api, gui.

13.1.1 Per system

SSH to a CCM node and on a command line execute following command:

```
% sbc-toggle-debug -c -e
```

This will enable debug logging of all supported tools in whole system. To disable debug logging just execute:

```
% sbc-toggle-debug -c
```

13.1.2 Per node

SSH to a node for which a debug level should be set and execute following commands:

```
% sbc-toggle-debug -a -e
```

This will enable debug logging of all supported tools for that specific node. To disable debug logging just execute:

```
% sbc-toggle-debug -a
```

Chapter 14

Reference of Call Agent Configuration Parameters

This reference lists all Call Agent configuration parameters used in ABC SBC. These parameters take effect on any traffic that is specific to a Call Agent without need to place any additional action into the Call Agent's rulebase.

The actions are grouped as follows:

- *Destination Monitor Parameters*
- *Blacklisting Parameters*
- *Registration Agent Parameters*
- *Topology Hiding Parameters*
- *Firewall Blacklisting Parameters*

14.1 Destination Monitor Parameters

These parameters can enable health checks for Call Agents. If enabled, along with blacklisting parameters unresponsive Call Agent's addresses will be proactively excluded from forwarding.

Parameter Name	Description
Monitoring interval (sec)	Interval between sending OPTIONS-based health-checks to the monitored Call Agent. If zero, no monitoring takes place.
Max-Forwards	Value of max-forwards header-field in the health-checking OPTIONS requests.

14.2 Blacklisting Parameters

These parameters allow to define for how long a CA's unresponsive address shall not be used for SIP traffic forwarding (i.e. how long it will be "blacklisted"). See the Section *IP Blacklisting: Adaptive Availability Management* for additional information.

Parameter Name	Description
Blacklist TTL (ms)	Period of time for which an unresponsive address is hold on blacklist. If zero, blacklisting is not used.
Blacklist grace timer (milliseconds)	Additional period of time to provide a safety buffer in case that conflicting timers occur along a SIP path.
Blacklist Reply Codes ABC Monitor	SIP Response codes which if present in a response will cause placing the downstream SIP server on the blacklist.

14.3 Registration Agent Parameters

Registration agent allows to register the ABC SBC with a third-party SIP service be sending pre-defined REGISTER requests as described in the Section *Registration Agent*. The following Call Agent parameters define if such a registration agent shall be active and how its registration parameters shall be formed.

Parameter Name	Description
Enabled	Turns a registration agent on or off.
URI domain.	Domain name to be used in REGISTER requests URIs
URI name.	User name to be used in REGISTER request URIs
Display name	Display names as included in the From header-field of the REGISTER requests
auth name	SIP User id as used in the authentication header fields. May be different from user names in URIs.
auth password	SIP user password used in the digest authentication
Contact	Content of the Contact header-field in the REGISTER requests. Specific usernames may be chosen to make it easier to identify incoming requests coming to addresses registered using the registration agent.
Contact HF Params	Semi-colon separated header parameters to add to the Contact header.
Registration interval (seconds)	Time between subsequent registrations are sent
Retry interval (seconds)	Period of time to keep till the next attempt when the previous failed
Next Hop (IP address)	Address of a destination to which a request will be sent
Bulk Contact	Turn on to support the SIP bulk contact registration form as described in RFC3680.

14.4 Topology Hiding Parameters

The Section *Topology Hiding* discussed purpose and use of Topology Hiding. The following options enable/disable this functionality for the respective Call Agents.

Parameter Name	Description
Enabled	Turning this option replaces occurrences of IP addresses in well-known header-fields of SIP signaling with those of the ABC SBC .
Cross-Realm	If enabled, topology hiding is used even when signaling ingress and egress realms are the same.

14.5 Firewall Blacklisting Parameters

Automated IP address blocking is discussed in the Section *Automatic IP Address Blocking*. Several attributes defined what kind of Call Agent behaviour adds to the score that may eventually lead to blacklisting of the source IP address.

Parameter Name	Description
Sanity	If turned on, invalid SIP messages add to the auto-blocking score and may lead to blocking of their originator. Otherwise they are silently ignored.
Auth	If enabled, failed authentication add to the auto-blocking score and may lead to blocking of their originator. Otherwise only events are reported but no further action is taken.

Chapter 15

Reference of Default Port Numbers

The reference lists port numbers the ABC SBC uses. It is particularly useful when considering firewall policies for firewalls placed in front of the ABC SBC. The reference lists default port numbers, transport protocols and the interface on which the respective applications are permitted. In addition to the SBC interfaces (see *SBC Interfaces*), some applications may be listening on all interfaces and some management applications are using the loopback interface for internal communication.

Note that while the ABC SBC only accepts traffic on the ports and interfaces specified in this specification, further restrictions may apply. Signaling is only accepted from well-defined Call Agents and certain traffic may be blacklisted (see *Manual SIP Traffic Blocking*).

Port	Transport	Description
22	ssh / TCP	Secure shell server. Used for remote management. Value 0 can be used for default port, which is 22 on standard SBC and 24 on container SBC. It can be set using ssh app on SBC interface. Until SBC is properly configured, the ssh default port 22 is opened from everywhere.
25	SMTP / TCP	Local Email relay. Used to forward email alerts. From outside perspective it acts as a client.
53	domain / TCP	Local DNS resolver. From outside perspective it acts as a client.
161	SNMP / UDP	Internal SNMP management.
443	https / TCP	Administrative GUI.
443	https / TCP	Tryit WebRTC client gui is accessible, if enabled, on default port and /tryit path.
1443	https / TCP	XMLRPC provisioning
3306	TCP	MariaDB configuration (UNIX sockets enforce)
5060	sip / UDP, TCP	SIP signaling
5061	sip / TLS	SIP signaling over TLS.
6379	TCP	redis replication, if HA is used
8080, 8081	TCP	SIP over Websocket WebRTC
8090	TCP	XML-RPC remote programming interface
10000 to 60000	UDP	Audio/video media.
15441, 4443	TCP	webconference demo available only on request
1444	TCP	RESTful port for AWS SNS, disabled by default
4242, 4243 4244, 4249	TLS	API for local monitoring data, queried from configuration master node.
4245, 4246	TCP	TCP port delivering various metrics and statistics about the host.
112	vrrp	If deployed in HA mode, the vrrp protocol advers are sent to node peer.

Additional fixed source port numbers shall be opened for the ABC SBC acting as client reaching outside servers as listed in the following table:

SBC Client Port	Description
NTP/123/UDP	Time Synchronization
domain/53/UDP	DNS Resolver

Other applications running on the ABC SBC use external applications while locally binding to ephemeral ports.

Remote Server Port	Description
HTTP/80	Software package updates
HTTPS/443	Software package updates
syslog/514	remote syslog facility if configured under Global Config / syslog-ng
rsync/873	remote PCAP/WAW storage if enabled under Global Config / replicate recordings / traffic log
rsync/1873	remote PCAP/WAW storage if enabled under Global Config / replicate recordings / traffic log, using TLS if secure connection to ABC Monitor enabled
https/444	config transfer from configuration master to ABC SBC nodes
6379,redis	redis replication and event generation to a ABC Monitor
16379,redis	redis replication and event generation to a ABC Monitor over TLS if enabled
SMTP/25	email alerts
ldaps/389	ldaps (ldap/tcp with STARTTLS) server

Chapter 16

Reference Interface Parameters

The following parameters can be defined at interface level:

Parameter Name	Description
force_via_address	When enabled, incoming requests are replied to the address shown in their Via header field. This conforms to the RFC3261 specification but often fails to traverse NATs and also permits a reflection attack through the ABC SBC.
wspath_xxx	The option, where xxx can be set as needed, sets up an HTTP proxy from path /xxx on HTTPS 443 port (or other port number if using a non-standard one on XMI) to the websocket port on localhost . (It has to be used only on interface using system interface “lo”.

Chapter 17

Reference Application Interface Options

Starting 4.5, the ABC SBC offers the possibility to configure some application option per logical interface, allowing a better control over which process is listening on which port.

Some applications require a TLS profile assigned to corresponding SBC or applied interface.

Initial available applications are:

- *SSH*
- *Management GUI*
- *Configuration pull*
- *Media*
- *Signaling*
- *WebSocket signaling*
- *SNMP*
- *XMLRPC*
- *TryIt webrtc client GUI*
- *TURN server for websocket*
- *PCAP query service*
- *Local monitoring query service*
- *Call state HA replication*

Starting 4.6, the following applications are also available:

- *Management for host*
- *HTTP proxy*
- *HTTP redirect*
- *Local webconf API*

In the following descriptions, those interfaces acronym stand for :

- *xmi*: external management interface
- *imi*: internal management interface
- *si*: signaling interface
- *mi*: media interface
- *ws*: websocket interface

- *ci*: custom interface

17.1 SSH

The ssh application allows a shell access via the associated interface on the configured port options.

The application may be enabled on all interface types, but by default it is enabled on *xmi* and *imi* only. Note: on freshly installed ABC SBC, until the *xmi* or *imi* interfaces are configured, the SSH is listening on all interfaces, using default port 22 (or 24 for SBC container).

Parameter Name	Description
Port	Port allowing ssh access.

17.2 Management GUI

The management GUI application enables https access to the CCM. Note that this application only has effect on CCM node and requires a TLS profile.

The application is exclusive and mandatory to *xmi* interface.

Parameter Name	Description
Port	Port allowing CCM web GUI https access.

17.3 Configuration pull

The configuration pull application allows SBC nodes to pull their configuration from the CCM node. Note that this application only has effect on CCM node. The configuration pull is using https protocol and requires a TLS profile.

The application is exclusive to *xmi* and *imi* interfaces and mandatory on the later one.

Parameter Name	Description
Port	Port allowing configuration pull. Note: value not editable (*444*).

17.4 Media

The media application impacts SBC communication handling. Note that this application only has effect on SBC node.

The application is exclusive and mandatory to *mi* interface.

The port range specifies a UDP port range used for media traffic, and does not use TLS.

Parameter Name	Description
Ports	Port range on which SBC may open a socket for media communications.
TOS	This sets “type of service” field in IP packets header.

17.5 Signaling

The signaling application impacts SBC communication handling. Note that this application only has effect on SBC node.

If "TLS Port" is not empty, a TLS profile is required.

The application is exclusive and mandatory to *si* interface.

Parameter Name	Description
Port	Ports on which SBC will open a signaling socket.
TLS Port	(optional) TLS port on which SBC opens a socket for secured signaling communication.
Interface Options	Special interface options. Note: allowed value is *force_via_address*.
TOS	This sets "type of service" field in IP packets header.
Greylist	Enables usage of greylist filter.

17.6 WebSocket signaling

The websocket application allows signaling communication over websocket interface.

If "TLS enabled" is set, a TLS profile is required.

The application is exclusive and mandatory to *ws* interface.

Parameter Name	Description
Port	Listening port of the websocket server.
TLS enabled	Enable secure communications.
Interface Options	Special interface options. Note: value must start by *wspath_*.
Greylist	Enables usage of greylist filter.

17.7 SNMP

The snmp application enables SNMP daemon listening. Note that this application only has effect on SBC node.

It does not require TLS profile, as TLS is not used.

The application may be enabled on *xmi* and *ci* interfaces.

Parameter Name	Description
Port	Port on which the SNMP server listens.

17.8 XMLRPC

The XMLRPC application enables SBC XMLRPC interface communication. Note that this application only has effect on CCM node.

It uses https protocol and requires a TLS profile.

The application is exclusive to *xmi* and *ci* interfaces.

Parameter Name	Description
Port	Listening port of the XMLRPC server.

17.9 TryIt webrtc client GUI

Enable the try-it application on SBC node, allowing WebRTC call from a web browser. It is possible to configure one TryIt application per node but it can be configured for more than one node.

It uses https protocol and requires a TLS profile.

The application may only be enabled on *ci* or *xmi* interfaces.

Parameter Name	Description
TLS Port	The https port of tryit web interface.
Path	The url path part on which tryit will be served.
websocket uri	The uri of websocket where the tryit should connect.
Whether the 'display name' is mandatory	Whether or not the 'display name' field on login screen of tryit is mandatory.
Minimum length of the 'display name'	Sets the min. length of display name allowed.
Prefix of the dialed SIP URIs	Prefix of the dialed SIP URIs. Example: +49308866019381.
Append random string to AOR	If checked the username in **From** header of INVITE/REGISTER is appended with a random string. The authentication username is kept untouched.
Separator of the random string in AOR	If **Append random string to AOR** is checked this is the separator between regular username and the random part.
SIP Domain	Domain part of the dialed SIP URIs. Example: mydomain.com. Note: required.
Client URL params base64 encoded	Whether or not are the GET params of client URL base64 encoded.
Keep-Alive timer	Interval for sending keepalive ping messages (in seconds). If zero or not set, no pings are sent. Note: required.
ICE url	The ICE Url that can set e.g. usage of TURN server.
ICE username	Username that will be used if "ICE url" is provided.
ICE credential	Password that will be used if "ICE url" is provided.
Audio media constraints	JSON encoded MediaTrackConstraints object providing the constraints which must be met by the outgoing audio track.
Video media constraints	JSON encoded MediaTrackConstraints object providing the constraints which must be met by the outgoing video track. Example: *{ width: { min: 320, ideal: 480, max: 1280}, height: { min: 240, ideal: 360, max: 720}, frameRate: { min: 10, max: 30 }}*.
URL of token authentication server	URL of token authentication server. If the URL contain string **{token}** it will be replaced by the token from client URL. Example: http://my.authserver.com/token={token}
Force TryIt to be refresher of session timer	In environment where the WebRTC client is connected via a client-side only initiated connection, it makes sense to force the client to be the refresher of session timer. All it takes is basically to add the **refresher=uac** parameter to the Session-Expires HF in the initial INVITE.
Allow insecure HTTPS connections	Whether or not allow insecure HTTPS connections to authentication server. Like connections with self signed certificate or so.

17.10 TURN server for websocket

It enables the TURN server on given node. It is possible to configure one TURN server per node but it can be configured for more than one node.

It does not require a TLS profile.

The application may be enabled on *ci* interface.

Parameter Name	Description
Listening port	Listening port of the TURN server.
Aux server	Auxiliary server address in the format *IP:port*.
Relay IP	Note: mandatory.
External IP	TURN Server public/private address mapping, if the server is behind NAT. In that situation, the External IP will be reported as relay IP address of all allocations. This scenario works only in a simple case when one single relay address is be used, and no RFC5780 functionality is required. That single relay address must be mapped by NAT to the 'external' IP. The External IP value, if not empty, is returned in XOR-RELAYED-ADDRESS field. For that 'external' IP, NAT must forward ports directly (relayed port 12345 must be always mapped to the same 'external' port 12345).
UDP port range min port	Sets the UDP range that is used for relaying media start port. Note: mandatory.
UDP port range max port	Sets the UDP range that is used for relaying media end port. Note: mandatory.
Auth user	Sets the username used for TURN server authentication. Note: mandatory.
Auth password	Sets the password used for TURN server authentication. Note: mandatory.
Realm for users	Realm passed, which is usually domain name.
Media IP to allow UDP on firewall	Sets the IP address that will be allowed on SBC firewall to talk to the TURN.

17.11 PCAP query service

The *sbc-pkapman* API generates and serves pcap files based on an aggregation of the pcap files available on the file system. The API will by default listen on the *localhost* interface, reachable via *http*. For every other interface application enabled, the API will listen exclusively via *https*, serving the configured TLS profile, which is required.

Requirements: sems's global option "Dump TLS session keys to file" *SEMS Parameters* must be enabled if one wishes to download both pcap files and session TLS keys into a zip'ed bundle. Otherwise, the bundle may only contain pcap files.

Limitations: WebRTC interface don't support dump of the TLS keys.

The application only exists on SBC node and it is mandatory and exclusive to *imi* interface.

Parameter Name	Description
Port	Port on which the API server listens. Note: value not editable (*4243*).

17.12 Local monitoring query service

The *sbc-xmloredis* API serves some metrics issued from different sources. The API will by default listen on the *localhost* interface, reachable via *http*. For every other interface application enabled, the API will listen exclusively via *https*, serving the configured TLS profile, which is required.

The application only exists on SBC node. It is also exclusive and mandatory to *imi* interface.

Parameter Name	Description
Port	Port on which the API server listens. Note: value not editable (*4242*).

17.13 Management for host

The *sbc-gominstrator* API

The API will by default listen on the *localhost* interface, reachable via *http*. For every other interface application enabled, the API will listen exclusively via *https*, serving the configured TLS profile, which is required.

The application may be enabled on *imi* interface.

Parameter Name	Description
Port	Port on which the API server listens. Note: value not editable (*4249*).

17.14 Local webconf API

The *sbc-webconf* API

The API will by default listen on the *localhost* interface, reachable via *http*. For every other interface application enabled, the API will listen exclusively via *https*, serving the configured TLS profile, which is required.

The application is exclusive and mandatory to *imi* interface.

Parameter Name	Description
Port	Port on which the API server listens. Note: value not editable (*4244*).

17.15 HTTP proxy

Setup an HTTP proxy, based on nginx reverse proxy. The application adds the *X-Real-IP*, *Upgrade* and *Connection* headers. The template (*/etc/frafos/templates/nginx/http_proxy.tmpl*) may be overloaded, as described in *Command Line Reference*.

If “TLS enable” is set, a TLS profile is required.

The application may be enabled on *ci* interface.

Parameter Name	Description
Source Port	Port from which the proxy should operate.
Source Path	Path from which the proxy should operate.
Target IP address	IP to which the proxy redirect. Note: mandatory.
Target port	Port to which the proxy redirect. Note: mandatory.
TLS enable	Proxy over TLS.

17.16 HTTP redirect

Setup an HTTP redirect pattern, using nginx rewrite directive.

The template (*/etc/frafos/templates/nginx/http_redirect.tmpl*) may be overloaded, as described in *Command Line Reference*.

If "TLS enable" is set, a TLS profile is required.

The application may be enabled on *ci* interface.

Parameter Name	Description
Port	Port from which the redirect should operate.
Path	Path from which the redirect should operate. Path is a regex to which we prefix ^ (start of line).
Target URL	URL to where be redirected. Note: mandatory.
TLS enable	Redirect over TLS.

17.17 Call state HA replication

Please note that the application have effect only if an HA is configured and used.

It uses internal redis protocol and does not require TLS profile.

The application is exclusive and mandatory for *imi* interface.

Parameter Name	Description
Port	Port on which call state redis will be listening. Note: value not editable (*6379*).

Chapter 18

Command Line Reference

The administrative GUI is the preferred way of the ABC SBC. However there are cases like the initial configuration and/or automation when accessing the ABC SBC via Command Line is useful.

18.1 Configuration Management

CLI	Purpose	Reference
sbc-install	initial ABC SBC installation	*Installation Procedure*
sbc-upgrade	ABC SBC upgrade	*Upgrade Procedure*
abc-monitor-install	initial ABC Monitor installation	*ABC Monitor Package Installation*
abc-monitor-upgrade	ABC Monitor upgrade	*ABC Monitor Package Installation*
sbc-backup	back up ABC SBC configuration	*ABC SBC Recovery Procedure*
sbc-restore	recovery of a backed up configuration	*ABC SBC Recovery Procedure*
sbc-set-confversion	forcibly sets config version number on config master	*ABC SBC Recovery Procedure*
sbc-init-config	This command configures IP address or DNS name of the main configuration node, from which ABC SBC node will automatically get configuration. It has to be run on all SBC nodes. This script is part of installation procedure.	*Web GUI Configuration (Choosing the Cluster Config Master)*
sbc-set-master	set up a configuration master	*Web GUI Configuration (Choosing the Cluster Config Master)*
sbc-publish-config	Activate the current SBC configuration and make it available for all nodes.	
sbc-daily-backup	Creates daily SBC backup, if enabled under Config / Global config / Backup tab.	
sbc-apply-config	Manually applies ABC SBC json configuration on slave node. Use –help option for command line options help.	
sbc-apply- provtables	Manually applies ABC SBC provisioned tables on slave. Use –help option for command line options help.	

18.2 User Management

CLI	Purpose	Reference
sbc-add-user	Add new GUI user or add a user to a group.	*CLI User Management*
sbc-del-user	Remove a GUI user or remove a user from a group.	*CLI User Management*
sbc-list-groups	Get list of existing user groups	*CLI User Management*
sbc-list-users	Get list of SBC users	*CLI User Management*
sbc-user-passwd	Change password of SBC user. Or unlock user locked by too many login attempts.	*CLI User Management*

18.3 Low-Level CLI

CLI	Purpose	Reference
sbc-create-config module	This command regenerates configuration files from their templates. Note: if needed, instead of tweaking the template itself (usually in /etc/frafos/templates), you must create a copy with the *.local* suffix. The *.tmpl.local* files have predominance over the *.tmpl* one.	
sbc-activate-config	like *sbc-create-config all* but restart the appropriate service after the config generation	
sbc-loglevel action [loglevel]	Shows or sets the logging level for the ABC SBC signalling process. Action is either 'get' to retrieve current value or 'set' to set it. Loglevel takes category and level. Log files are stored in the directory /var/log/frafos	*Reference of Log Level Parameters*
sbc-status	Shows ABC SBC node status, which is collected automatically every minute and also shown on config master node GUI on Nodes status page.	
sbc-init-services	Initial services initialization done during installation.	
sbc-events-queue	Show number of events waiting in redis queue on Sbc to be delivered to primary and secondary ABC Monitor.	

18.4 HA CLI

In previous ABC SBC releases up to 4.1, the high availability solution used was based on Pacemaker. The ABC SBC 4.2 was a transitional release that removed the Pacemaker based HA solution, before new Keepalived based HA solution was introduced in 4.3 release.

CLI	Purpose	Reference
sbc-ha-offline	Forces the node when run to be put forcibly into HA FAULT state	
sbc-ha-online	Clears the forcibly set HA FAULT state set by sbc-ha-offline	
sbc-ha-status	Shows the node's current HA status, which can be MASTER, SLAVE or FAULT.	

18.5 ABC Monitor Configuration Management

CLI	Purpose	Reference
abc-monitor-backup-config	create backup of ABC Monitor config	*ABC Monitor Backup and Restore Operations*
abc-monitor-restore-config	restore backup of ABC Monitor config	*ABC Monitor Backup and Restore Operations*
abc-monitor-set-gui	configures ABC Monitor GUI https port	

Chapter 19

Reference of Used Open-Source Software

The key components of ABC SBC are built as commercial software fully owned by FRAFOS GmbH and its subsidiaries. Additionally it relies on the Linux operating systems and numerous accompanying libraries and components provided by third parties under the following license terms:

- bash , GPLv3+
- boost: Boost Software License & similar (http://www.boost.org/users/license.html)
- cronie , MIT and BSD and ISC and GPLv2+
- crontabs , Public Domain and GPLv2
- dialog , LGPLv2
- dmidecode , GPLv2+
- ethtool , GPLv2
- expat (XML parser): MIT https://sourceforge.net/p/expat/code_git/ci/master/tree/expat/COPYING
- fail2ban, GPLv2
- fence-agents-all , GPLv2+ and LGPLv2+
- flite , X11-like http://www.festvox.org/flite/doc/flite_2.html
- hiredis , BSD https://github.com/redis/hiredis/blob/master/COPYING
- iLBC: BSD-like
- js , GPLv2+ or LGPLv2+ or MPLv1.1
- json-c: MIT (https://github.com/json-c/json-c/blob/master/COPYING)
- jsonxx: MIT? (https://github.com/hjiang/jsonxx/blob/master/LICENSE)
- libbcg729: GPLv3 (https://github.com/BelledonneCommunications/bcg729/blob/master/LICENSE.txt)
- libcap , LGPLv2+
- libcurl: MIT/X derivate license https://curl.haxx.se/docs/copyright.html
- libevent: BDS-like http://libevent.org/LICENSE.txt
- libisac: WebRTC license
- libopus: BSD
- libosip2 , LGPLv2+
- libpcap , BSD with advertising
- librsvg2 , LGPLv2+
- libsrtp , BSD-like https://github.com/cisco/libsrtp/blob/master/LICENSE

- libtiff , BSD-like (http://www.libtiff.org/misc.html)
- libwebsockets , LGPL2.1 https://github.com/warmcat/libwebsockets/blob/master/LICENSE
- libxml2 , MIT http://www.xmlsoft.org/FAQ.html
- mailx , BSD with advertising and MPLv1.1
- mariadb-server , GPLv2 with exceptions and LGPLv2 and BSD
- monit , AGPLv3
- mysql++ , LGPLv2
- mysql-connector-c++ , GPLv2 with exceptions
- MySQL-python , GPLv2+
- nginx, BSD-like
- net-snmp , BSD http://www.net-snmp.org/about/license.html
- net-snmp-utils , BSD
- ntp , (MIT and BSD and BSD with advertising) and GPLv2
- opencore-amr: Apache V2.0
- openssh-clients , BSD
- openssl, BSD-like https://www.openssl.org/source/license.html
- opus , BSD
- pciutils , GPLv2+
- pcmisc , GPLv2+
- pcs , GPLv2
- perl-Net-SSLeay , OpenSSL
- php-cli , PHP and Zend and BSD
- php-db , PHP
- php-log , PHP
- php-mysql , PHP
- php-pear-XML-RPC , PHP
- php-pecl-runkit , PHP
- php-xmlrpc , PHP and BSD
- python , Python
- python-jinja2 , BSD
- redis , BSD
- rsync , GPLv3+
- sems-gsm , public domain
- sems-speex , modified BSD
- serweb-frmwrk , GPL
- silk: BSD-like
- spandsp (g722, DTMF): LGPL
- speex , BSD
- sqlite , Public Domain

- stunnel, GPL
- syslog-ng , GPLv2+
- sysstat , GPLv2+
- tcpdump , BSD with advertising
- vconfig , GPLv2+
- yajl (JSON): ISC license https://en.wikipedia.org/wiki/ISC_license
- wireshark , GPL+

The ABC SBC comes also packaged with a JavaScript JSSIP.net demo application for WebRTC telephony. The application has been adopted by FRAFOS for demonstration purposes and comes with no support. The copyright holder is José Luis Millán and the software is available under the MIT license.

The ABC Monitor is based on the "ELK" stack which consists of the following components:

- logstash, Apache2 License
- Elastic Search, Apache2 License
- kibana, Apache License
- nginx, BSD-like
- redis , BSD

Chapter 20

Reference Userdata Parameters for AWS Instances

The behaviour of the ABC SBC can be altered by Userdata passed to it during instance launch. See the following link for more information about Userdata: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html#instancedata-add-user-data

The ability to alter the instance behaviour is often useful when instances are started using a CloudFormation template. The parameters passed through Userdata must be encoded as attribute name:value pair; name and value are separated by comma and so are the pairs.

The following table shows reserved attribute names and how they are used.

Attribute Name and Value	Description
configurl <URL>	Download an ABC SBC backup configuration (only applicable when ismater:TRUE, if the instance is slave it retrieves the configuration from its master.
cwgroup <NAME>	an additional CloudWatch Dimension to which the ABC SBC sends CloudWatch metrics; this can be used to group metrics from multiple instances; note that proper CloudWatch permissions must be set
cwregion <REGION>	if CloudWatch metrics is to be gathered in a different region than instance's own, set the CloudWatch region using this parameter
ismaster TRUE	Enforce configuration master role
master <IP address>	Run this instance as configuration slave of a master identified by an IP address.
remotebootscript <URL>	URL of a bash script that will be downloaded and sourced during instance launch. The script must be finite because the boot process doesn't continue until it completes.
rtcecdns <IP address>	Address of the primary Monitor

Note that any attribute names including custom ones can be passed via Userdata. When a remotebootscript is used and started, all the attributes are passed to it as shell variables.

An example of UserData may look like this:

```
rtcecdns,172.12.1.1, configurl, https://s3-eu-west-1.amazonaws.com/frafos-
↪abcconfig/40014-honeypot.sql
```

Chapter 21

Reference XML-RPC functions

In the case that the ABC SBC administrator needs to configure large data sets to CCM GUI, it will be easier to provision those data automatically with a script as opposed to typing it in using the web-interface. This can be accomplished using the ABC SBC's XML-RPC data provisioning interface.

The following example shows a python code fragment for accessing the built-in XML-RPC provisioning server:

```
#!/usr/bin/python
import xmlrpclib;
server = xmlrpclib.Server('https://username:password@10.0.0.10:1443/rpc.php');
```

Note that for the python client, a question mark (?) in the password does not work. The user accessing XML-RPC interface has to be either member of **SBCrpc** group or member of another group having **XML-RPC** privilege.

For the XML-RPC access the IP address of the configuration master node has to be used. Also the *XMLRPC* application has to be enabled for the particular interface. The XML-RPC is accessible by default on port 1443.

The XML-RPC interface is self documented via function *rpc.help()*. When the function is called without any argument it prints list of all available function. When function name is given as an argument to this function (*rpc.help(<function name>)*) it will return detailed help of the specified function.

For example try following calls in python:

```
print server.rpc.help()
print server.rpc.help('rpc.help')
```

As of now functions for manipulate following entities are available:

- *Provisioned Tables*
- *Call agents*
- *TLS profiles*
- *Nodes*
- *Logical interfaces*
- *System interfaces*

Bellow is list of all available XML RPC functions. Call *rpc.help(<function name>)* to get detailed help of specified function.

21.1 Provisioned Tables

Functions for define provisioned tables and manipulate data in them.

Function Name	Description
tables.fetch_rules($table_name)	Get all rules from specified provisioned table.
tables.fetch_rule($table_name, $key_values)	Get a rule matching the key from the specified provisioned table.
tables.insert_rule($table_name, $data)	Insert rule into specified provisioned table.
tables.insert_rules($table_name, $rules)	Insert multiple rules into specified provisioned table.
tables.update_rule($table_name, $data)	Update rule of specified provisioned table.
tables.insert_update_rule($table_name, $data)	Try update rule of specified provisioned table. If rule with matching UUID or key columns does not exists, new rule is inserted.
tables.delete_rule($table_name, $uuid)	Delete rule from specified provisioned table.
tables.delete_all_rules($table_name)	Delete all rules from specified provisioned table.
tables.commit($table_name, $msg)	Commit working version of provisioned table into use by signaling and create new working version by copying the current one.
tables.fetch()	Get all provisioned table definitions.
tables.insert($payload)	Insert provisioned table.
tables.update($payload)	Update provisioned table.
tables.delete($table_name)	Delete provisioned table.

For example, to introduce a new entry to the blacklist and check the outcome, the following three RPC commands must be called: *insert_rule*, *commit* and *fetch_rules*:

```
data = {"key_value":"sip:restricted@abc.com"};
print server.tables.insert_rule('test_uri_bl',data);
print server.tables.commit('test_uri_bl', 'new restricted used introduced');
print server.tables.fetch_rules('test_uri_bl');
```

This script will result in the following list of URIs shown on the command-line output:

```
[{'key_value': 'sip:banned@abcsbc.com', 'uuid': '6c01a834-9d32-df09-0217-
↪000000f074ee'},
{'key_value': 'sip:forbidden@abcsbc.com', 'uuid': '54d15a12-62bc-73c9-8313-
↪000012f8ae1b'},
{'key_value': 'sip:restricted@abc.com', 'uuid': '6d831a12-88bc-7fa9-7483-
↪000083ff992a'}]
```

Note that the routing tables have several predefined mandatory elements that must use the following conventions:

- *cagent* takes name or UUID of a call-agent
- *outbound_proxy* and *next_hop* is passed as string
- boolean parameters *next_hop_1st_rq*, *upd_ruri_host*, and *upd_ruri_dns_ip* take either 0 or 1 as value
- the enumerative parameters *route_via* takes one of the following values: *outbound_proxy*, *next_hop* or *ruri*

21.2 Call agents

Function Name	Description
cagents.fetch($filter)	Get call agents
cagents.insert($payload)	Insert call agent
cagents.update($payload)	Update call agent
cagents.delete($realm_name, $cagent_name)	Delete call agent
cagents.add_target($realm_name, $cagent_name, $payload)	Add target destination to call agent
cagents.del_target($realm_name, $cagent_name, $payload)	Remove target destination from call agent

21.3 TLS profiles

Function Name	Description
tls_profile.fetch($filter)	Get TLS profiles
tls_profile.insert($payload)	Insert TLS profile
tls_profile.update($payload)	Update TLS profile
tls_profile.delete($name)	Delete TLS profile

21.4 Nodes

Function Name	Description
node.fetch($filter)	Get SBC nodes
node.insert($payload)	Insert SBC node
node.update($payload)	Update SBC node
node.delete($name)	Delete SBC node

21.5 Logical interfaces

Function Name	Description
log_interface.fetch($filter)	Get logical interfaces
log_interface.insert($payload)	Insert logical interface
log_interface.update($payload)	Update logical interface
log_interface.delete($name)	Delete logical interface
log_interface.help_app_list()	Return list of available applications
log_interface.help_app($application)	Return detailed info about an application

21.6 System interfaces

Function Name	Description
sys_interface.fetch($filter)	Get system interfaces
sys_interface.insert($payload)	Insert system interface
sys_interface.update($payload)	Update system interface
sys_interface.delete($log_if_name, $owner_type, $owner_name)	Delete system interface

Chapter 22

Glossary

3GPP	3rd Generation Partnership Project
AoR	Address of Record
B2BUA	Back to Back User Agent
BLF	Busy Lamp Field
CA	Call Agent
CDR	Call Data Record / Call Detail Record
CPU	Central Processing Unit
DNS	Domain Name System
DoS	Denial of Service
ENUM	Electronic Number Mapping System
FQDN	Fully Qualified Domain Name
HA	High availability
IMI	Internal Management Interface
IMS	IP Multimedia Subsystem
IP	Internet Protocol
ISUP	ISDN User Part
MI	Media Interface
NAT	Network Address Translator
NIC	Network Interface Card
NNI	Network-Network Interface
PBX	Private Exchange
PSTN	Public Switched Telecommunication Network
REST	Representational state transfer
RLM	Realm
RTP	Real-Time Transport Protocol
RTCP	Real-Time Transport Control Protocol
SAP	Session Announcement Protocol
SBC	Session Border Controller
SCTP	Stream Control Transport Protocol
SDP	Session Description Protocol
SI	Signalling interface
SIP	Session Initiation Protocol
SNMP	Simple Network Management Protocol
SST	SIP Session Timers
SRV	Service Record
STUN	Session Traversal Utilities for NAT
TISPAN	Telecommunications and Internet converged Services and Protocols for Advanced Networking
TCP	Transport Control Protocol

continues on next page

Table 1 – continued from previous page

TLS	Transport Level Security
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UNI	User-Network Interface
URI	Universal Resource Indicator
VIP	Virtual IP Address
VoIP	Voice over IP
XMI	External Management Interface

Index

R

RFC
    RFC 1889, 7
    RFC 2327, 7
    RFC 2617, 104
    RFC 2782, 94
    RFC 2833, 110
    RFC 3261, 7, 87, 107, 111, 131, 275
    RFC 3263, 23, 26, 88, 94
    RFC 3264, 112
    RFC 3325, 101, 106, 275
    RFC 3581, 10
    RFC 3608, 275
    RFC 3711, 161
    RFC 3761, 158
    RFC 3960, 107
    RFC 4028, 264
    RFC 4122, 148
    RFC 4145, 118
    RFC 4244, 111, 275
    RFC 4347, 161
    RFC 4474, 159
    RFC 4733, 110
    RFC 5242, 161
    RFC 5245, 13
    RFC 5359, 108
    RFC 5389, 13, 161
    RFC 5628, 10
    RFC 5766, 13
    RFC 5806, 111, 275
    RFC 5853, 10
    RFC 6044, 111
    RFC 6062, 161
    RFC 6140, 131, 156
    RFC 6386, 161
    RFC 6716, 161
    RFC 7118, 161