Zorp Professional 7 Administrator Guide

Publication date March 04, 2024

Abstract

This document is the primary guide for Zorp Professional administrators.

Table of Contents

Preface ... **xiv**
1. Summary of contents ... xiv
2. Target audience and prerequisites ... xv
3. Products covered in this guide ... xvi
4. Contact and support information ... xvi
4.1. Sales contact ... xvii
4.2. Support contact ... xvii
4.3. Training ... xvii
5. About this document ... xvii
5.1. Feedback ... xvii
Summary of changes ... **xviii**
1. Introduction ... **1**
1.1. What Zorp is ... 1
1.2. Who uses Zorp? ... 2
2. Concepts of the Zorp Gateway solution ... **3**
2.1. Main components of the Zorp Gateway solution ... 3
2.1.1. Zorp ... 4
2.1.2. Zorp Management Server (ZMS) ... 4
2.1.3. Transfer Agent ... 5
2.1.4. Zorp Management Console (ZMC) ... 5
2.1.5. Zorp Authentication Server (ZAS) ... 6
2.1.6. The concept of the ZCV framework ... 6
2.1.7. Virtual Private Networking (VPN) support ... 10
2.1.8. Native services ... 10
2.1.9. High Availability ... 11
2.1.10. Operating system ... 11
2.2. The concepts and architecture of Zorp firewalls ... 11
2.2.1. Access control ... 11
2.2.2. Operation modes of Zorp ... 11
2.2.3. Packet filtering in Zorp ... 12
2.2.4. Proxying connections ... 12
2.2.5. Traffic analysis with proxies ... 13
2.2.6. Proxy customization ... 14
2.2.7. Modular architecture ... 14
3. Managing Zorp hosts ... **15**
3.1. ZMS and ZMC ... 15
3.1.1. Defining a new host and starting ZMC ... 16
3.2. ZMC structure ... 17
3.2.1. Configuration tree ... 18
3.2.2. Main workspace ... 23
3.2.3. Menu & status bars and Preferences ... 24
3.3. Configuration and Configuration management ... 31
3.3.1. Configuration process ... 31
3.3.2. Configuration buttons ... 32
3.3.3. Committing related components ... 39

3.3.4. Recording and commenting configuration changes 39
3.3.5. Multiple access and lock management 40
3.3.6. Status indicator icons 42
3.3.7. Copy, paste and multiple select in ZMC 45
3.3.8. Links and variables 46
3.3.9. Disabling rules and objects 47
3.3.10. Filtering list entries 48
3.4. Viewing Zorp logs 51
3.4.1. The command bar of the log viewer 52
4. Registering new hosts 53
4.1. Bootstrap a new host 53
4.2. Reconnecting to a host 61
4.2.1. Reconnecting ZMS to a host 61
5. Networking, routing, and name resolution 63
5.1. Configuring networking interfaces 64
5.1.1. General interface configuration 64
5.1.2. Configuring virtual networks and alias interfaces 68
5.1.3. Configuring bond interfaces 71
5.1.4. Configuring bridge interfaces 71
5.1.5. Enabling spoof protection 72
5.1.6. Interface options and activation scripts 73
5.1.7. Interface status and statistics 78
5.2. Managing name resolution 80
5.3. Managing client-side name resolution 81
5.3.1. Configure name resolution 81
5.4. The routing editor 83
5.4.1. Routes 84
5.4.2. Sorting, filtering, and disabling routes 85
5.4.3. Managing the routing tables locally 86
6. Managing network traffic with Zorp 87
6.1. Understanding Zorp policies 87
6.2. Zones 87
6.2.1. Managing zones with ZMC 88
6.2.2. Creating new zones 90
6.2.3. Zone hierarchies 93
6.2.4. Using hostnames in zones 96
6.2.5. Finding zones 98
6.2.6. Exporting zones 98
6.2.7. Importing zones 100
6.2.8. Deleting a zone or more zones simultaneously 102
6.3. Zorp instances 104
6.3.1. Understanding Zorp instances 104
6.3.2. Managing Zorp instances 105
6.3.3. Creating a new instance 107
6.3.4. Configuring instances 108
6.3.5. Instance parameters — general 109
6.3.6. Instance parameters — logging 111
6.3.7. Instance parameters — Rights 112

6.3.8. Instance parameters — miscellaneous ... 113
6.3.9. Increasing the number of running processes ... 114
6.4. Zorp services ... 115
6.4.1. Creating a new service ... 115
6.4.2. Creating a new packet filtering Service (PFService) ... 118
6.4.3. Creating a new DenyService ... 119
6.4.4. Creating a new DetectorService ... 120
6.4.5. Routing — selecting routers and chainers ... 122
6.5. Configuring firewall rules ... 135
6.5.1. Understanding Zorp firewall rules ... 135
6.5.2. Transparent and non-transparent traffic ... 139
6.5.3. Finding firewall rules ... 139
6.5.4. Creating firewall rules ... 140
6.5.5. Tagging firewall rules ... 144
6.5.6. Configuring nontransparent rules with inband destination selection ... 145
6.5.7. Connection rate limiting ... 146
6.6. Proxy classes ... 147
6.6.1. Customizing proxies ... 148
6.6.2. Renaming and editing proxy classes ... 158
6.6.3. Analyzing embedded traffic ... 159
6.7. Policies ... 160
6.7.1. Creating and managing policies ... 162
6.7.2. Detector policies ... 162
6.7.3. Encryption policies ... 164
6.7.4. Matcher policies ... 165
6.7.5. NAT policies ... 173
6.7.6. Resolver policies ... 182
6.7.7. Stacking providers ... 184
6.8. Monitoring active connections ... 186
6.9. Traffic reports ... 188
6.9.1. Configuring Zorp reporting ... 188
7. Logging with syslog-ng ... 190
7.1. Introduction to syslog-ng ... 191
7.1.1. Global options ... 191
7.1.2. Sources ... 191
7.1.3. Destinations ... 192
7.1.4. Filters ... 193
7.2. Configuring syslog-ng with ZMC ... 194
7.2.1. Configure syslog-ng ... 194
7.2.2. Configuring syslog-ng components through ZMC ... 195
7.2.3. Configuring TLS-encrypted logging ... 209
8. The Text editor plugin ... 213
8.1. Using the Text editor plugin ... 213
8.1.1. Configure services with the Text editor plugin ... 213
8.1.2. Use the additional features of Text editor plugin ... 215
9. Native services ... 218
9.1. BIND ... 218
9.1.1. BIND operation modes ... 219

9.1.2. Configuring BIND with ZMC 219
9.1.3. Setting up split-DNS configuration 222
9.2. NTP 223
9.2.1. Configuring NTP with ZMC 223
9.2.2. Status and statistics 226
9.3. Postfix 227
9.3.1. Configuring Postfix with ZMC 228
9.4. Local services on Zorp 232
9.4.1. Enabling access to local services 233
10. Local firewall administration 235
10.1. Linux 235
10.2. Login to the firewall 236
10.3. Editing configuration files 237
10.4. Network configuration 238
10.5. System logging 239
10.6. NTP 240
10.7. BIND 241
10.8. Updating and upgrading your Zorp hosts 242
10.9. Packet filter 243
10.10. Zorp configuration 244
10.10.1. Policy.py and instances.conf 245
10.10.2. Zorp control 247
10.11. Managing core dump files 248
11. Key and certificate management in Zorp 249
11.1. Cryptography basics 249
11.1.1. Symmetric and asymmetric encryption 249
11.2. PKI Basics 252
11.2.1. Centralized PKI system 252
11.2.2. Digital certificates 253
11.2.3. Creating and managing certificates 253
11.2.4. Verifying the validity of certificates 255
11.2.5. Verification of certificate revocation state 255
11.2.6. Authentication with certificates 256
11.2.7. Digital encryption in work 256
11.2.8. Storing certificates and keys 257
11.3. PKI in ZMS 257
11.3.1. Committing changes and locking in PKI 258
11.3.2. The certificate entity 258
11.3.3. Rules of distribution and owner hosts 258
11.3.4. Trusted groups 259
11.3.5. The PKI menu 259
11.3.6. PKI management 263
11.3.7. Trusted CAs 267
11.3.8. Managing certificates 273
12. Clusters and high availability 288
12.1. Introduction to clustering 288
12.2. Clustering solutions 288
12.2.1. Fail-Over clusters 288

12.2.2. Load balance clusters 290
12.3. Managing clusters with ZMS 292
12.4. Creating clusters 293
12.4.1. Creating a new cluster (bootstrapping a cluster) 294
12.4.2. Adding new properties to clusters 296
12.4.3. Adding a new node to a Zorp cluster 297
12.4.4. Converting a host to a cluster 299
12.5. Heartbeat 299
12.5.1. Functionality of Heartbeat 299
12.5.2. Heartbeat resources 301
12.5.3. Configuring Heartbeat 301
12.5.4. Configuring Heartbeat resources 308
12.5.5. Configuring a Service IP address 309
12.6. Keepalived for High Availability 310
12.6.1. Functionality of Keepalived 310
12.6.2. Prerequisites for configuring Keepalived 310
12.6.3. Configuring Keepalived 310
12.6.4. Configuration examples and best practices for Keepalived configuration 317
12.7. Availability Checker 319
12.7.1. Prerequisites for configuring the Availability Checker plugin 319
12.7.2. 319
13. Advanced ZMS and Agent configuration 323
13.1. Setting configuration parameters 324
13.1.1. Configuring user authentication and privileges 325
13.1.2. Configuring backup 330
13.1.3. Configuring the connection between ZMS and ZMC 332
13.1.4. Configuring ZMS and agent connections 335
13.1.5. Configuring ZMS database save 337
13.1.6. Setting configuration check 338
13.1.7. Configuring CRL update settings 338
13.1.8. Set logging level 339
13.1.9. Configuring SSL handshake parameters 340
13.2. Setting agent configuration parameters 342
13.2.1. Configuring connections for agents 343
13.2.2. Configuring connection to engine 343
13.2.3. Configuring logging for agents 343
13.2.4. Configuring SSL handshake parameters for agents 344
13.3. Managing connections 346
13.3.1. Setting up initial connection with management agents 346
13.3.2. Configuring connection with agents 346
13.3.3. Administering connections 347
13.3.4. Configuring recovery connections 348
13.4. Handling XML databases 349
14. Virus and content filtering using ZCV 351
14.1. Content vectoring basics 351
14.1.1. Quarantining 352
14.2. Content vectoring with ZCV 352
14.2.1. Creating module instances 352

14.2.2. Creating scanpaths ... 364
14.2.3. Routers and rule groups ... 370
14.2.4. Configuring Zorp proxies to use ZCV ... 375
14.2.5. Managing ZCV performance and resource use ... 381
14.3. Quarantine management in ZMC ... 384
14.3.1. Information stored about quarantined objects ... 385
14.3.2. Configuring quarantine cleanup ... 386
15. Connection authentication and authorization ... 389
15.1. Authentication and authorization basics ... 389
15.1.1. Inband authentication ... 389
15.1.2. Outband authentication ... 390
15.2. The concept of ZAS ... 391
15.2.1. Supported backends and authentication methods ... 392
15.3. Authenticating connections with ZAS ... 392
15.3.1. Configuring ZAS ... 392
15.3.2. Authentication of Zorp services with ZAS ... 401
15.3.3. Authorization of Zorp services ... 411
15.3.4. Enabling Kerberos authentication in ZAS ... 419
15.3.5. Configuring the authentication agent ... 424
15.4. Logging in ZAS ... 424
16. Virtual Private Networks ... 426
16.1. Virtual Private Networking basics ... 426
16.1.1. Types of VPN ... 427
16.1.2. VPN topologies ... 427
16.1.3. The IPSec protocol ... 427
16.1.4. The OpenVPN protocol ... 428
16.2. Using VPN connections ... 428
16.2.1. Using VPN connections ... 429
16.3. Configuring IPSec connections ... 430
16.3.1. Configuring IPSec connections ... 430
16.3.2. IPSec options ... 438
16.3.3. Global IPSec options ... 441
16.3.4. Forwarding IPSec traffic on the packet level ... 441
16.4. Configuring SSL (OpenVPN) connections ... 441
16.4.1. Prerequisities for configuring SSL (OpenVPN) connections ... 441
16.4.2. Configuring SSL connections ... 442
16.4.3. SSL options ... 450
17. Integrating Zorp to external monitoring systems ... 459
17.1. Monitoring Zorp with Munin ... 459
17.2. Installing a Munin server on a ZMS host ... 459
17.3. Monitoring Zorp with Nagios ... 460
Appendix A. Packet Filtering ... 462
A.1. How packet filtering works ... 462
A.2. Packet filtering on Linux ... 463
A.3. Understanding Netfilter and IPTables ... 464
A.3.1. Hooks ... 464
A.3.2. Tables ... 466
A.3.3. Chains ... 468

A.3.4. Rules 469
A.3.5. Configuration summary 470
A.4. Managing packet filter rules in ZMC 471
A.4.1. Configuration management: iptables-utils 472
A.4.2. Modifying the ruleset 473
A.4.3. Understanding the packet filter ruleset 474
A.4.4. The Rule Search window 481
Appendix B. Keyboard shortcuts in Zorp Management Console 483
B.1. Function keys 483
B.2. Shortcuts 483
B.3. Access keys 484
Appendix C. Further readings 485
C.1. Zorp-related material 485
C.2. General, Linux-related materials 485
C.3. Postfix documentation 485
C.4. BIND Documentation 485
C.5. NTP references 485
C.6. SSH resources 485
C.7. TCP/IP Networking 485
C.8. Netfilter/IPTables 486
C.9. General security-related resources 486
C.10. syslog-ng references 486
C.11. Python references 486
C.12. Public key infrastructure (PKI) 486
C.13. Virtual Private Networks (VPN) 486
Appendix D. Zorp Professional End-User License Agreement 487
D.1. 1. SUBJECT OF THE LICENSE CONTRACT 487
D.2. 2. DEFINITIONS 487
D.3. 3. LICENSE GRANTS AND RESTRICTIONS 488
D.4. 4. SUBSIDIARIES 490
D.5. 5. INTELLECTUAL PROPERTY RIGHTS 490
D.6. 6. TRADE MARKS 490
D.7. 7. NEGLIGENT INFRINGEMENT 490
D.8. 8. INTELLECTUAL PROPERTY INDEMNIFICATION 490
D.9. 9. LICENSE FEE 491
D.10. 10. WARRANTIES 491
D.11. 11. DISCLAIMER OF WARRANTIES 492
D.12. 12. LIMITATION OF LIABILITY 492
D.13. 13.DURATION AND TERMINATION 492
D.14. 14. AMENDMENTS 492
D.15. 15. WAIVER 493
D.16. 16. SEVERABILITY 493
D.17. 17. NOTICES 493
D.18. 18. MISCELLANEOUS 493
Appendix E. Creative Commons Attribution Non-commercial No Derivatives (by-nc-nd) License 494

List of Examples

3.1. Referring to components with variables 47
5.1. Referencing static and dynamic interfaces in firewall rules 68
6.1. Using the Internet zone 89
6.2. Subnetting 90
6.3. Finding IP networks 98
6.4. Customized logging for HTTP accounting 112
6.5. Overriding the target port SQLNetProxy 125
6.6. Overriding the target port SQLNetProxy 128
6.7. RFC-compliant proxying in Zorp 148
6.8. Virus filtering and stacked proxies 159
6.9. Defining a Detector policy 163
6.10. DNSMatcher for two domain names 167
6.11. Defining a RegexpMatcher 169
6.12. Blacklisting e-mail recipients 172
6.13. SmtpProxy class using a matcher for controlling relayed zones 173
6.14. Address translation examples using **GeneralNAT** 178
6.15. Defining a Resolver policy 183
6.16. Using HashResolver to direct traffic to specific servers 184
7.1. Selecting log messages from Postfix using filter 205
7.2. Setting up a router 208
9.1. Forward-only DNS server 219
9.2. Split-DNS implementation 219
9.3. Special requirements on mail handling 227
10.1. Specifying the target IP address of a TCP destination 240
15.1. BasicAccessList 417
A.1. Chaining 469
A.2. Protection against spoof 479

List of Procedures

2.1.6.1. Content vectoring with ZCV 9
3.1.1. Defining a new host and starting ZMC 16
3.2.1.3.1. Adding new configuration components to host 22
3.2.3.1. Configuring general ZMC preferences 24
3.2.3.2. Configuring Zorp Class Editor preferences 27
3.2.3.3. Configuring Zorp Rules preferences 28
3.2.3.4. Configuring ZMS hosts 28
3.2.3.6.1. Defining variables 30
3.2.3.6.2. Editing variables 30
3.2.3.6.3. Deleting variables 30
3.3.1.1. Configuring Zorp - the general process 31
3.3.4. Recording and commenting configuration changes 39
4.1. Bootstrap a new host 53
4.2.1. Reconnecting ZMS to a host 61
5.1.1.1. Configuring a new interface 65
5.1.2.1. Creating a VLAN interface 69
5.1.2.2. Creating an alias interface 70
5.1.3. Configuring bond interfaces 71
5.1.4. Configuring bridge interfaces 71
5.1.5.1. Configuring spoof protection 72
5.1.6.1.1. Creating interface activation scripts 74
5.1.6.2.1. Creating interface groups 76
5.1.6.3.1. Configuring interface parameters 78
5.3.1. Configure name resolution 81
5.4.2.1. Filtering routes 85
6.2.2. Creating new zones 90
6.2.3.1. Organizing zones into a hierarchy 94
6.2.6. Exporting zones 98
6.2.7. Importing zones 100
6.2.8. Deleting a zone or more zones simultaneously 102
6.3.3. Creating a new instance 107
6.3.4. Configuring instances 108
6.3.9. Increasing the number of running processes 114
6.4.1. Creating a new service 115
6.4.2. Creating a new packet filtering Service (PFService) 118
6.4.3. Creating a new DenyService 119
6.4.4. Creating a new DetectorService 120
6.4.5.1. Setting routers and chainers for a service 122
6.5.3. Finding firewall rules 139
6.5.4. Creating firewall rules 140
6.5.5. Tagging firewall rules 144
6.5.7. Connection rate limiting 146
6.6.1.1. Derive a new proxy class 149
6.6.1.2. Customizing proxy attributes 150
6.6.2. Renaming and editing proxy classes 158

6.6.3.1. Stack proxies 160
6.7.1. Creating and managing policies 162
6.7.5.1.1. Configuring NAT 175
6.9.1. Configuring Zorp reporting 188
7.2.1. Configure syslog-ng 194
7.2.2.1.1. Set global options 196
7.2.2.2.1. Create sources 200
7.2.2.2.2. Create drivers 200
7.2.2.4.1. Set filters 205
7.2.2.5.1. Configure routers 207
7.2.3. Configuring TLS-encrypted logging 209
8.1.1. Configure services with the Text editor plugin 213
8.1.2. Use the additional features of Text editor plugin 215
9.1.2.1. Configuring BIND with ZMC 219
9.1.3. Setting up split-DNS configuration 222
9.2.1. Configuring NTP with ZMC 223
9.3.1.1. Configuring Postfix with ZMC 228
9.4.1. Enabling access to local services 233
10.8. Updating and upgrading your Zorp hosts 242
10.10.1.1. Edit the Policy.py file 245
11.1.1.4.1. Procedure of encrypted communication and authentication 251
11.2.3.1. Creating a certificate 254
11.3.7.2. Creating a new CA 270
11.3.7.4. Signing CA certificates with external CAs 273
11.3.8.2. Creating certificates 276
11.3.8.3. Revoking a certificate 280
11.3.8.4. Deleting certificates 282
11.3.8.5. Exporting certificates 282
11.3.8.6. Importing certificates 284
11.3.8.7. Signing your certificates with external CAs 286
11.3.8.8. Monitoring licenses and certificates 286
12.4.1. Creating a new cluster (bootstrapping a cluster) 294
12.4.2. Adding new properties to clusters 296
12.4.3. Adding a new node to a Zorp cluster 297
12.4.4. Converting a host to a cluster 299
12.5.3.1. Configure Heartbeat 301
12.5.3.2. Configure additional Heartbeat parameters 305
12.5.4. Configuring Heartbeat resources 308
12.5.5. Configuring a Service IP address 309
12.6.3.1. Configure Keepalived 310
12.6.4.1. Simple Cluster with 2 nodes 317
12.6.4.2. Testing or Pilot node 317
12.6.4.3. Multiple backup nodes 317
12.6.4.4. Multiple VRRP groups in the same cluster 317
12.6.4.5. Managing individual OpenVPN tunnels 318
12.7.2.1. Configuring the Availability Checker 319
13.1.1.1. Adding new users to ZMS 326
13.1.1.2. Deleting users from ZMS 327

13.1.1.3. Changing passwords in ZMS 327
13.1.1.4.1. Editing user privileges in ZMS 328
13.1.1.5.1. Modifying authentication settings 329
13.1.2.1. Configuring automatic ZMS database backups 330
13.1.2.2. Restoring a ZMS database backup 331
13.1.3.1. Configuring the bind address and the port for ZMS-ZMC connections 332
13.1.3.2. Using linking for the IP address 335
13.1.4. Configuring ZMS and agent connections 335
13.1.5. Configuring ZMS database save 337
13.1.8. Set logging level 339
13.1.9. Configuring SSL handshake parameters 340
13.2.3. Configuring logging for agents 343
13.2.4. Configuring SSL handshake parameters for agents 344
13.3.3. Administering connections 347
13.3.4. Configuring recovery connections 348
14.2.1.1. Creating a new module instance 353
14.2.2.1. Creating a new scanpath 364
14.2.3.1. Creating and configuring routers 372
14.2.4.1. Configuring communication between Zorp proxies and ZCV 375
15.1.2.1. Outband authentication using the Zorp Authentication Agent 390
15.3.1.1.1. Creating a new instance 393
15.3.2.1. Configuring communication between Zorp and ZAS 401
15.3.2.2. Configuring Zorp Authentication policies 406
15.3.3.1. Configuring authorization policies 411
15.3.4. Enabling Kerberos authentication in ZAS 419
16.2.1. Using VPN connections 429
16.3.1. Configuring IPSec connections 430
16.3.4. Forwarding IPSec traffic on the packet level 441
16.4.2. Configuring SSL connections 442
16.4.3.1. Configuring the VPN management daemon 452
17.1. Monitoring Zorp with Munin 459
17.2. Installing a Munin server on a ZMS host 459
17.3. Monitoring Zorp with Nagios 460
A.4.4.1. Using Rule Search 481

Preface

Welcome to the Zorp Professional 7 Administrator Guide!

This document describes how to configure and manage Zorp Professional 7 and its components. Background information for the technology and concepts used by the product is also discussed.

1. Summary of contents

Chapter 1, Introduction (p. 1) describes the main functionality and purpose of the Zorp Professional.

Chapter 2, Concepts of the Zorp Gateway solution (p. 3) describes the features and capabilities of the different components of Zorp, as well as the concepts of Zorp.

Chapter 3, Managing Zorp hosts (p. 15) describes the main configuration utility of Zorp.

Chapter 4, Registering new hosts (p. 53) explains how to manage several firewalls using a single management server.

Chapter 5, Networking, routing, and name resolution (p. 63) describes the management of network interfaces, such as Ethernet cards.

Chapter 6, Managing network traffic with Zorp (p. 87) describes how to customize the firewall system for optimal security.

Chapter 7, Logging with syslog-ng (p. 190) introduces the capabilities of syslog-ng.

Chapter 8, The Text editor plugin (p. 213) discusses how to manage external services from Zorp Management Console.

Chapter 9, Native services (p. 218) describes the built-in DNS, NTP and mailing services of Zorp.

Chapter 10, Local firewall administration (p. 235) explains how to manage Zorp from a local console.

Chapter 11, Key and certificate management in Zorp (p. 249) introduces the use and management of certificates.

Chapter 12, Clusters and high availability (p. 288) introduces the use and management of Zorp clusters.

Chapter 13, Advanced ZMS and Agent configuration (p. 323) discusses various advanced topics.

Chapter 14, Virus and content filtering using ZCV (p. 351) discusses the concepts, configuration, and use of the Zorp Content Vectoring System framework and the related modules.

Chapter 15, Connection authentication and authorization (p. 389) details the authentication and authorization services provided by Zorp and the Zorp Authentication Server.

Chapter 16, Virtual Private Networks (p. 426) how to build encrypted connections between remote networks and hosts using virtual private networks (VPNs).

Chapter 17, Integrating Zorp to external monitoring systems (p. 459) describes how to integrate Zorp to the monitoring infrastructure.

Appendix A, Packet Filtering (p. 462) discusses how to configure and manage the built-in packet filter of Zorp.

Appendix B, Keyboard shortcuts in Zorp Management Console (p. 483) describes the keyboard shortcuts available in Zorp Management Console.

Appendix C, Further readings (p. 485) is a list of suggested reference materials in different Zorp and network security related fields.

Appendix D, Zorp Professional End-User License Agreement (p. 487) includes the text of the End-User License Agreement applicable to Zorp products.

Appendix E, Creative Commons Attribution Non-commercial No Derivatives (by-nc-nd) License (p. 494) includes the text of the Creative Commons Attribution Non-commercial No Derivatives (by-nc-nd) License applicable to The Zorp Professional 7 Administrator Guide.

2. Target audience and prerequisites

This guide is intended for use by system administrators and consultants responsible for network security and whose task is the configuration and maintenance of Zorp firewalls. Zorp gives them a powerful and versatile tool to create full control over their network traffic and enables them to protect their clients against Internet-delinquency.

This guide is also useful for IT decision makers evaluating different firewall products because apart from the practical side of everyday Zorp administration, it introduces the philosophy behind Zorp without the marketing side of the issue.

The following skills and knowledge are necessary for a successful Zorp administrator.

Skill	**Level/Description**
Linux	At least a power user's knowledge is required.
Experience in system administration	Experience in system administration is certainly an advantage, but not absolutely necessary.
Programming language knowledge	It is not an explicit requirement to know any programming languages though being familiar with the basics of Python may be an advantage, especially in evaluating advanced firewall configurations or in troubleshooting misconfigured firewalls.
General knowledge on firewalls	A general understanding of firewalls, their roles in the enterprise IT infrastructure and the main concepts and tasks associated with firewall administration is essential. To fulfill this requirement a significant part of *Chapter 3, Architectural overview* in the *Zorp Administrator's Guide* is devoted to the introduction to general firewall concepts.

Skill	Level/Description
Knowledge on Netfilter concepts and IPTables	In-depth knowledge is strongly recommended; while it is not strictly required definitely helps understanding the underlying operations and also helps in shortening the learning curve.
Knowledge on TCP/IP protocol	High level knowledge of the TCP/IP protocol suite is a must, no successful firewall administration is possible without this knowledge.

Table 1. Prerequisites

3. Products covered in this guide

The Zorp Distribution DVD-ROM contains the following software packages:

- Current version of Zorp 7 packages.
- Current version of Zorp Management Server (ZMS) 7.
- Current version of Zorp Management Console (ZMC) 7 (GUI) for both Linux and Windows operating systems, and all the necessary software packages.
- Current version of Zorp Authentication Server (ZAS) 7.
- Current version of the Zorp Authentication Agent (ZAA) 7, the ZAS client for both Linux and Windows operating systems.

For a detailed description of hardware requirements of Zorp, see *Chapter 1, System requirements* in *Zorp Professional 7 Installation Guide*.

For additional information on Zorp and its components visit the *Zorp website* containing white papers, tutorials, and online documentations on the above products.

4. Contact and support information

This product is developed and maintained by Balasys IT Zrt..

Contact:

Balasys IT Zrt.
4 Alíz Street
H-1117 Budapest, Hungary
Tel: +36 1 646 4740
E-mail: `<info@balasys.hu>`
Web: *http://balasys.hu/*

4.1. Sales contact

You can directly contact us with sales related topics at the e-mail address `<sales@balasys.hu>`, or *leave us your contact information and we call you back*.

4.2. Support contact

To access the Balasys Support System, sign up for an account at *the Balasys Support System page*. Online support is available 24 hours a day.

Balasys Support System is available only for registered users with a valid support package.

Support e-mail address: `<support@balasys.hu>`.

4.3. Training

Balasys IT Zrt. holds courses on using its products for new and experienced users. For dates, details, and application forms, visit the *https://www.balasys.hu/en/services#training* webpage.

5. About this document

This guide is a work-in-progress document with new versions appearing periodically.

The latest version of this document can be downloaded from *https://docs.balasys.hu/*.

5.1. Feedback

Any feedback is greatly appreciated, especially on what else this document should cover, including protocols and network setups. General comments, errors found in the text, and any suggestions about how to improve the documentation is welcome at `<support@balasys.hu>`.

Summary of changes

The following changes have been made to the document between releases Zorp 7.0.19 and Zorp 7.0.20:

Description of the change	Place in the document
Changes to a zone in the ZMC, such as creating, modifying or deleting zones are made visible immediately in the same ZMC. After comitting the changes to a zone, the modifications are also made visible for other ZMCs.	For details, see *Section 6.2, Zones (p. 87)*.

Table 2. Summary of Changes

The following changes have been made to the document between releases Zorp 7.0.18 and Zorp 7.0.19:

Description of the change	Place in the document
For achieving an unobstructed installation of Zorp, a warning is placed in the document to make sure that OpenVPN is already installed.	For details, see *Section 16.4, Configuring SSL (OpenVPN) connections (p. 441)*.
The platform information on ZMC installation can be found in Zorp Professional Installation Guide.	For details, see section *Chapter 5, Installing the Zorp Management Console* in *Zorp Professional 7 Installation Guide*. Also see section *Section 2.1.4, Zorp Management Console (ZMC) (p. 5)*.

Table 3. Summary of Changes

The following changes have been made to the document between releases Zorp 7.0.17 and Zorp 7.0.18:

Description of the change	Place in the document
Kerberos authentication is enabled in ZAS using Active Directory.	For details, see *Procedure 15.3.4, Enabling Kerberos authentication in ZAS (p. ?)*.

Table 4. Summary of Changes

The following changes have been made to the document between releases Zorp 7.0.16 and Zorp 7.0.17:

Description of the change	**Place in the document**
When configuring keepalived HA cluster, setting the *keep-configuration* option prevents from manually configured IPs and routes being dropped at the restart of the Networking component.	For details, see *Procedure 12.6.3.1, Configure Keepalived (p. 310)*

Table 5. Summary of Changes

Chapter 1. Introduction

This chapter introduces the Zorp Professional (Zorp) in a non-technical manner, discussing how and why it is useful, and what additional security it offers to an existing IT infrastructure.

1.1. What Zorp is

Zorp provides complete control over regular and encrypted network traffic, with the capability to filter and also modify the content of the traffic.

Zorp is a perimeter defense tool, developed for companies with extensive networks and high security requirements. Zorp inspects and analyzes the content of the network traffic to verify that it conforms to the standards of the network protocol in use (for example, HTTP, IMAP, and so on). Zorp provides central content filtering including virus- and spamfiltering at the network perimeter, and is capable of inspecting a wide range of encrypted and embedded protocols, for example, HTTPS and POP3S used for secure web browsing and mailing. Zorp offers a central management interface for handling multiple firewalls, and an extremely flexible, scriptable configuration to suit divergent requirements.

The most notable features of Zorp are the following:

Complete protocol inspection: In contrast with packet filtering firewalls, Zorp handles network connections on the proxy level. Zorp ends connections on one side, and establishes new connections on the other; that way the transferred information is available on the device in its entirety, enabling complete protocol inspection. Zorp has inspection modules for over twenty different network protocols and can inspect 100% of the commands and attributes of the protocols. All proxy modules understand the specifications of the protocol and can reject connections that violate the standards. Also, every proxy is capable to inspect the TLS- or SSL-encrypted version of the respective protocol.

Unmatched configuration possibilities: The more parameters of a network connection are known, the more precise policies can be created about the connection. Complete protocol inspection provides an immense amount of information, giving Zorp administrators unprecedented accuracy to implement the regulations of the security policy on the network perimeter. The freedom in customization helps to avoid bad trade-offs between effective business-processes and the required level of security.

Reacting to network traffic: Zorp cannot only make complex decisions based on information obtained from network traffic, but is also capable of modifying certain elements of the traffic according to its configuration. This allows to hide data about security risks, and can also be used to treat the security vulnerabilities of applications protected by the firewall.

Controlling encrypted channels: Zorp offers complete control over encrypted channels. The thorough inspection of embedded traffic can in itself reveal and stop potential attacks like viruses, trojans, and other malicious programs. This capability of the product provides protection against infected e-mails, or websites having dangerous content, even if they arrive in encrypted (HTTPS, POP3S, or IMAPS) channels. The control over SSH and SSL traffic makes it possible to separately handle special features of these protocols, like port- and x-forwarding. Furthermore, the technology gives control over which remote servers can the users access by verifying the validity of the server certificates on the firewall. That way the company security policy can deny access to untrusted websites having invalid certificates.

Centralized management system: The easy-to-use, central management system provides a uniform interface to configure and monitor the elements used in perimeter defense: Zorp devices, content vectoring servers, as well as clusters of these elements. Different, even completely independent groups of Zorp devices can be managed from the system. That way devices located on different sites, or at different companies can be administered using a single interface.

Content vectoring on the network perimeter: Zorp provides a platform for antivirus engines. Using Zorp's architecture, these engines become able to filter data channels they cannot access on their own. Zorp's modularity and its over twenty proxy modules enable virus- and spamfiltering products to find malicious content in an unparalleled number of protocols, and their encrypted versions.

Single Sign On authentication: Linking all network connections to a single authentication greatly simplifies user-privilege management and system audit. Zorp's single sign on solution is a simple and user-friendly way to cooperate with Active Directory. Existing LDAP, PAM, AD, and RADIUS databases integrate seamlessly with Zorp's authentication module. Both password-based and strong (S/Key, SecureID, X.509, and so on) authentication methods are supported. X.509-based authentication is supported by the RDP and SSH proxies as well, making it possible to use smartcard-based authentication mechanisms and integrate with enterprise PKI systems.

1.2. Who uses Zorp?

The protection provided by the Zorp application-level perimeter defense technology satisfies even the highest security needs. The typical users of Zorp come from the governmental, financial, and telecommunication sectors, including industrial companies as well. Zorp is especially useful in the following situations:

- to protect networks that handle sensitive data or provide critical business processes
- to solve unique, specialized IT security problems
- to filter encrypted channels (for example, HTTPS, POP3S, IMAPS, SMTPS, FTPS, SFTP, and so on)
- to perform centralized content filtering (virus and spam) even in encrypted channels
- to filter specialized protocols (for example, Radius, SIP, SOAP, SOCKS, MS RPC, VNC, RDP, and so on)

Chapter 2. Concepts of the Zorp Gateway solution

This chapter provides an overview of the Zorp Gateway solution, introduces its main concepts and explains the relationship of the various components.

2.1. Main components of the Zorp Gateway solution

A typical Zorp Gateway solution consists of the following components:

- One or more **Zorp firewall hosts**. Zorp is inspecting and analyzing all connections.
- A **Zorp Management Server (ZMS)**
  ZMS is the central managing server of the Zorp Gateway solution. ZMS stores the settings of every component, and generates the configuration files needed by the other components. A single ZMS can manage the configuration of several Zorp firewalls — for example, if an organization has several separate facilities with their own firewalls, each of them can be managed from a central Zorp Management Server.
- One or more desktop computers running the **Zorp Management Console (ZMC)**, the graphical user interface of ZMS
  The Zorp administrators use this application to manage the entire system.
- **Transfer agents**
  These applications perform the communication between ZMS and the other components.
- One or more **Zorp Content Vectoring System (ZCV) servers**
  ZCV servers can inspect and filter the content of the network traffic, for example, using different virus- and spamfiltering modules. ZCV can inspect over 10 network protocols, including encrypted ones as well. For example, SMTP, HTTP, HTTPS, and so on.
- One or more **Zorp Authentication Server (ZAS)**
  ZAS can authenticate every network connection of the clients to a variety of databases, including LDAP, RADIUS, or TACACS. Clients can also authenticate out-of-band using a separate authentication agent.

Note
The name of the application effectively serving as the Zorp component of Zorp Professional is Zorp, commands, paths and internal references will relate to that naming.

The following figure shows how these components operate:

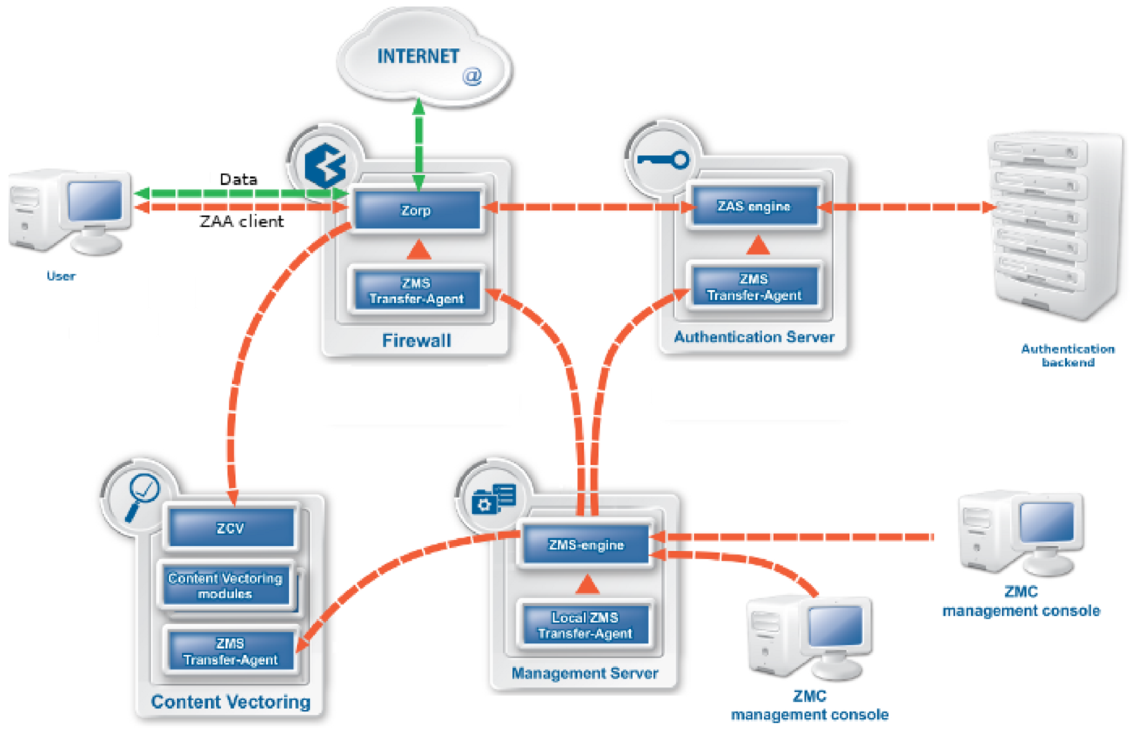


Figure 2.1. The architecture of the Zorp firewall system

2.1.1. Zorp

The heart of the Zorp-based firewall solution is the firewalling software itself, which is a set of proxy modules acting as application layer gateways. Zorp is an application proxy firewall. For details on the architecture of Zorp itself, see *Section 2.2, The concepts and architecture of Zorp firewalls (p. 11)*.

Zorp must be installed on an Ubuntu-based operating system (Ubuntu 18.04 LTS) which installs automatically when booting from the Zorp installation media.

2.1.2. Zorp Management Server (ZMS)

The Zorp Management Server (ZMS) handles the configuration tasks of the entire solution. To access ZMS and modify the configuration of the firewalls, the Zorp Management Console (ZMC) application on the desktop can be used. ZMS is the central command center of the solution: it stores and manages the configuration of Zorp firewall hosts.

The real power of ZMS surfaces when more than one Zorp firewalls have to be administered: instead of configuring the different firewalls individually and manually, it is possible configure them at a central location with ZMS, and upload the configuration changes to the firewalls. As ZMS stores the configuration of every firewall, the configuration of the entire firewall system can be backed up. In case of an emergency, it is possible to restore the configuration of every firewall with a few clicks.

2.1.3. Transfer Agent

Technically, ZMS does not communicate directly with the Zorp host: all communication is done through the Zorp Transfer Agent application, which is responsible for transporting configuration files to the managed hosts, running ZMS-initiated commands, and reporting the firewall configuration and other related information to ZMS. The Zorp Transfer Agent is automatically installed on every Zorp host. The communication is secured using Secure Socket Layer (SSL) encryption. The communicating hosts authenticate each other using certificates. For more information, see *Section 13.1.1.5, Configuring authentication settings in ZMS (p. 329)*.

Communication between the agents and ZMS uses TCP port `1311`. If Zorp and ZMS are installed on the same host, the communication between the transfer agent and the ZMS server uses UNIX domain sockets.

Warning
Agent connections must be enabled on every managed host, otherwise ZMS cannot control the hosts. For details, see *Appendix A, Packet Filtering (p. 462)*.

By default, the ZMS host initiates the communication channel to the agents, but the agents can also be configured to start the communication, if required.

2.1.4. Zorp Management Console (ZMC)

The Zorp Management Console (ZMC) is the graphical interface to Zorp Management Server (ZMS). A single ZMS engine can manage several different Zorp firewalls. ZMC is designed so, that almost all administration tasks of Zorp can be accomplished with it and therefore no advanced Linux skills are required to manage the firewall.

Note
ZMC can connect to the ZMS host remotely, even over the Internet. All connections between ZMC and ZMS are SSL-encrypted, and use TCP port `1314`.

ZMC can only alter configurations stored in the ZMS database. It does not directly communicate with the firewall hosts.

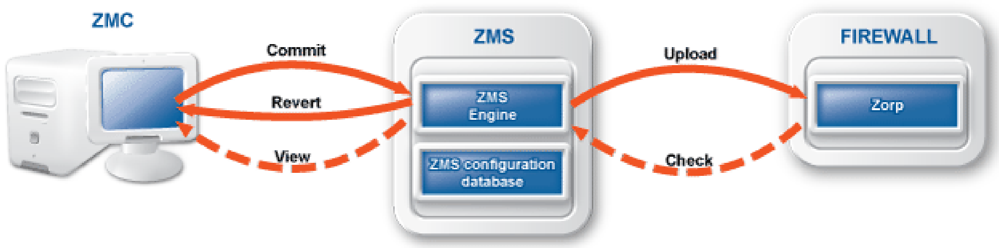


Figure 2.2. Communication between ZMC, ZMS, and Zorp

For information on the installation of ZMC see Chapter 5, Installing the Zorp Management Console in *Zorp Professional 7 Installation Guide*.

2.1.5. Zorp Authentication Server (ZAS)

Zorp can authenticate every connection: it is a single sign-on (SSO) authentication point for network connections. During authentication, Zorp communicates with the Zorp Authentication Agent (ZAA) application that runs on the client computers.

However, Zorp does not have database access for authentication information such as usernames, passwords and access rights. It operates indirectly with the help of authentication backends through an authentication middleware, the Zorp Authentication Server (ZAS). To authenticate a connection, Zorp connects to ZAS, and ZAS retrieves the necessary information from a user database. ZAS notifies Zorp about the results of the authentication, together with some additional data about the user that can be used for authorization.

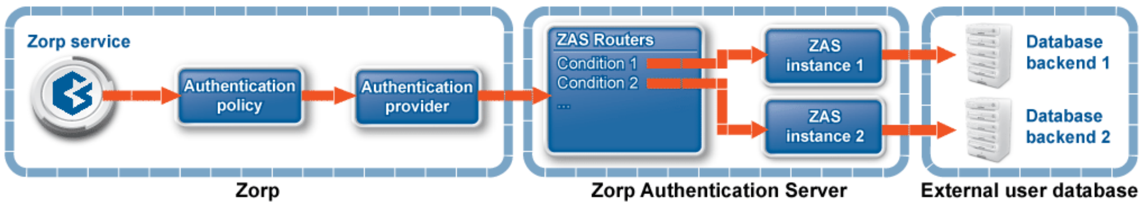


Figure 2.3. The operation of ZAS

ZAS supports the following user database backends:

- plain file in Apache htpasswd format
- Pluggable Authentication Module (PAM) framework
- RADIUS server
- LDAP server (plain BIND, password authentication, or with own LDAP scheme)
- Microsoft Active Directory

ZAS supports the following authentication methods:

- plain password-based authentication
- challenge/response method (S/KEY, CryptoCard RB1)
- X.509 certificates
- Kerberos 5

2.1.6. The concept of the ZCV framework

ZCV is not a content vectoring engine, it is a framework to manage and configure various third-party content vectoring modules (engines) from a uniform interface. Zorp uses these modules to filter the traffic. These modules run independently from Zorp. They do not even have to run on the same machines. Zorp can send the data to be inspected to these modules, along with configuration parameters appropriate for the scenario. For example, a virus filtering module can be used to inspect all files in the traffic, but different parameters can be used to inspect files in HTTP downloads and e-mail attachments. Also, different scenarios can use a different set of modules for inspecting the traffic. Using the above example, HTTP traffic can be inspected with a virus filter, a content filter, and all client-side scripts can be removed. E-mails can be scanned for viruses using the same virus filtering module (but possibly with stricter settings), and also inspected by a spam filtering module.

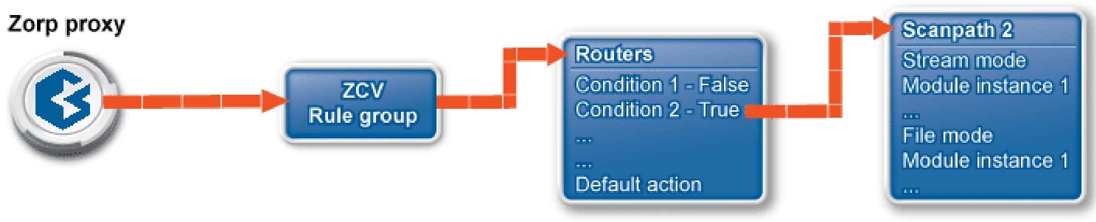


Figure 2.4. Interaction of Zorp and ZCV

The interaction of Zorp and ZCV takes place as follows:

- A Zorp proxy can send data for further inspection to a ZCV *rule group*.
- A *rule group* is used to define a scenario (using a set of *router rules*).
- The *router rules* of the scenario are condition – action pairs that determine how a particular object should be inspected. This decision is based on meta-information about the traffic or objects received from Zorp and on information collected by ZCV.
  - The condition can be any information that Zorp/ZCV can parse, for example, the client's IP address, the MIME-type of the object, and so on.
  - The action is either a default action (such as `ACCEPT` or `REJECT`), or a *scanpath* — a list of content vectoring *module instances* (the modules and their settings corresponding to the scenario) that will inspect the traffic. *Rule groups* have a *scanpath* configured as default, but the routers in the group can select a different *scanpath* for certain conditions.

The examples demonstrated on *Figure 2.5, Content vectoring scenarios in ZCV (p. 8)* can be translated to the ZCV terms defined in the previous paragraph as follows:

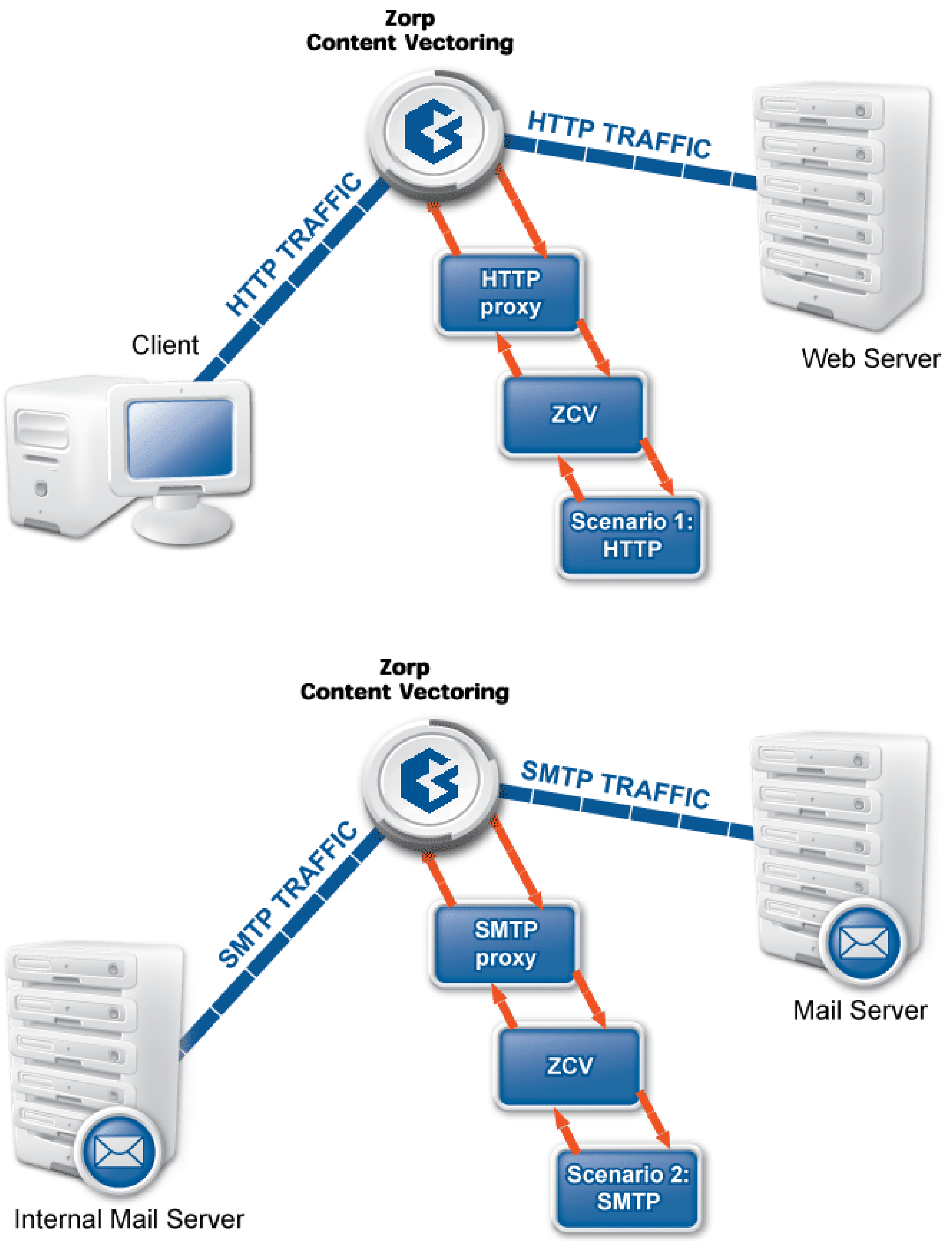


Figure 2.5. Content vectoring scenarios in ZCV

1. There are two *rule groups* (scenarios) defined, one for HTTP traffic, one for SMTP.
2. *The Router rules* formulate a *scanpath* in the HTTP rule group.

3. The *scanpath* includes *module instances* of a virus filtering, a content filtering, and an HTML module that are configured to remove all scripts.

This is only a basic example, but further *router rules* can be used to optimize the decisions. For example, it is unreasonable to remove client-side scripts in non-HTML files that are downloaded, and so on. Similarly, another *rule group* corresponds to the SMTP scenario, with a *scanpath* including a virus filtering and a spam filtering *module instance*.

The whole process is summarized in the following procedure.

2.1.6.1. Procedure – Content vectoring with ZCV

Step 1. A Zorp proxy sends the traffic to be inspected to an appropriate ZCV rule group.

Step 2. ZCV evaluates the router conditions of the rule group. If no condition is fulfilled, the action set as default (a default scanpath, or an `ACCEPT/REJECT`) is performed. Otherwise, the action/scanpath specified for the condition is followed.

Step 3. The traffic is inspected by the module instances specified in the selected scanpath. A module instance can be used in multiple scanpaths, with different parameters in each one.

Step 4. The processed traffic is returned to Zorp.

2.1.6.2. Supported modules

The ZCV framework was designed to allow the easy and fast integration of various third-party content vectoring tools. Currently the following modules are supported:

- **Virus filtering modules**:
  - *NOD32* (*http://www.eset.com*)
  - *Clam AntiVirus* (*http://clamav.net*)
- **Spam filtering modules**:
  - *SpamAssassin* (*http://spamassassin.apache.org/*)
- **Other modules**
  - *HTML filtering module*: It is capable of general content filtering, as well as filtering JavaScript, ActiveX, Java, and Cascading Style Sheets (CSS).
  - *Mail header filtering*: It is capable of filtering and manipulating mail headers in SMTP traffic.
  - *Mime filtering*: It is capable of filtering, removing, and adding MIME headers and objects in HTTP, POP3, and SMTP traffic.
  - *General stream editing module*: It is capable of replacing strings in data streams.
  - *Custom application*: ZCV provides a general interface to inspect the data with other third-party applications.

Some of the listed modules must be licensed separately from Zorp/ZCV. For details, contact your distributor.

2.1.7. Virtual Private Networking (VPN) support

Zorp uses strongSwan to support native Linux IPSec solutions, and also supports OpenVPN (an SSL-based VPN solution). Zorp supports both transport and tunnel mode VPN channels. Tunnel authentication is possible with X.509 certificates and with pre-shared keys (PSK). IPSec settings can be negotiated manually, or by using Internet Key Exchange (IKE).

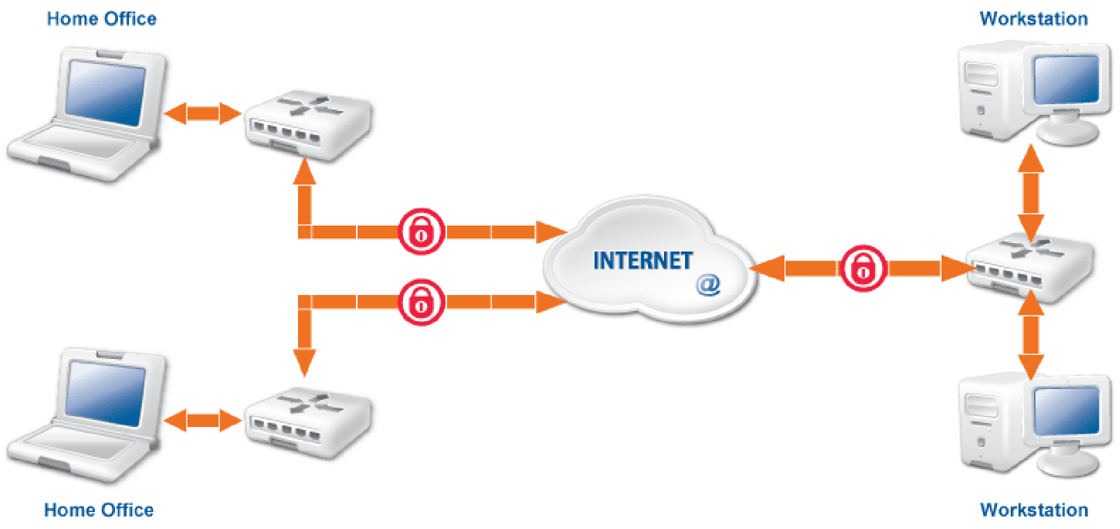


Figure 2.6. Virtual Private Networks

2.1.8. Native services

Native services provide a limited number of server-like features in Zorp. Their use is optional and depends on the needs and security requirements of your organization. The use of Network Time Protocol (NTP) and Bind is recommended, while Postfix is useful for managing mail traffic from various firewall components locally.

These services are called native because they are installed with Zorp and are available by default. They are implementations of the actual Linux services of the same names. These services provide networking services that are either difficult to implement with application proxies (or at the packet filter level) or provide services for the firewall itself. For more information on these services, see *Chapter 9, Native services (p. 218)*.

- **NTP**: Zorp hosts can function both as a Network Time Protocol (NTP) client and server. Time synchronization among the Zorp hosts is very important for correct logging entries. Once the firewall's time is correctly synchronized, it can act as the authentic time source for its internal networks.
- **DNS**: Zorp features a fully functional ISC BIND 9 DNS server. It is optional and definitely not mandatory to use if security regulations explicitly prohibit the installation of non-firewall software on the firewall machine. However, in small and mid-sized networks, it can be beneficial to have a built-in name server, if it is solely used as a forward–only DNS server.
- **SMTP**: Zorp uses Postfix as the built-in SMTP server component. Postfix is used for SMTP queuing. Zorp also has an application proxy for inspecting SMTP traffic, while ZCV can be used to perform virus, spam, and content-based filtering on the SMTP traffic. The primary role of this Postfix service is to provide a Mail Transport Agent (MTA) for the firewall itself: a number of mail messages can originate from the firewall, and these messages are delivered using the Postfix service. Although the

Postfix service is a fully functional MTA in Zorp, it is not intended to be a general purpose mail server solution for any organization.

2.1.9. High Availability

Zorp supports multi-node (2+) failover clustering, as well as load balance clusters (most load balance configurations use external devices). Clustering support must be licensed separately. Zorp supports the following failover methods:

- Transferring the Service IP address
- Transferring IP and MAC address
- Sending RIP messages

For more information see *Chapter 12, Clusters and high availability (p. 288)*.

2.1.10. Operating system

Zorp runs on Ubuntu-based operating systems. Currently it supports Ubuntu 18.04 LTS. The Zorp packages can either be installed on an existing Ubuntu server installation from the official Balasys APT repositories, or the installation media can be used to install a minimal Ubuntu 18.04 LTS server and the Zorp packages.

2.2. The concepts and architecture of Zorp firewalls

The following sections discuss the main concepts of Zorp firewalls.

2.2.1. Access control

A firewall controls which networks and hosts can be accessed, and who can access them. To create traffic rules, first, the networking environment of Zorp must be accurately defined, then, access control on the traffic can be applied. This can be achieved using zones and rules.

Zones consist of one or more IP subnets that Zorp handles together. By default, there is only a single zone: the IP network `0.0.0.0/0`, which practically means every available IP addresses (that is, the entire Internet). It is possible to organize zones into a hierarchy to reflect the actual network, or the structure of the organization.

Although zones consist of IP subnets and/or individual IP addresses, zone organization is independent of the subnetting practices of the organization. For example, a zone can be defined that contains the `192.168.7.0/24` subnet and it can have a subzone with IP addresses from the `10.0.0.0/8` range, and the single IP address of `172.16.54.4/32`. For details on zones, see *Section 6.2, Zones (p. 87)*.

2.2.2. Operation modes of Zorp

The first line of network defense is a packet filter that blocks traffic based on the IP address or TCP/UDP port number of the source (that is, the client) or the destination (that is, the server) of the connection. That way, more thorough analysis, such as traffic inspection or content vectoring is performed only on traffic that is permitted at all. This technology using both packet filtering and application proxying together is called multilayer filtering.

- **Packet filtering**: Traffic that can be filtered based on IP and TCP/UDP header information can be blocked at the packet filter level. Likewise, it is possible to forward traffic at the packet filter level without analyzing them with application proxies. For such traffic, Zorp operates like an ordinary packet filtering firewall. Forwarding traffic at the packet filter level may be desirable special protocols that cannot be proxied, or if proxying causes performance problems in the connection, or in case of non-TCP/UDP or bulk traffic. Zorp provides a number of built-in, protocol-specific proxy classes for the most typical protocols, and it has a generic proxy for protocols not supported by the built-in proxies. Packet filter level forwarding is not recommended, unless it is absolutely unavoidable.
  Application proxies provide a higher level of security. Packet filters are the first line of defense, they can be used to block unwanted traffic. What is not blocked by default, on the other hand, should be filtered by the appropriate application proxies.

- **Traffic proxying**: Application-level services inspect the traffic on protocol level (Layer 7 in the OSI model). Zorp provides a generic proxy, called PlugProxy that does not perform special data analysis, but can be used to proxy the traffic. Application proxies always provide an additional level of filtering over packet filters.

2.2.3. Packet filtering in Zorp

In Zorp, packet filtering is handled by the *kzorp* kernel module, therefore packet filtering services are completely handled on the kernel level. When Zorp starts, it sends all information about the traffic permitted to pass the gateway (that is, the list of configured services, zones, firewall rules, and so on) to the kzorp module.

Application-level services (also called proxy services) are handled on two levels:

- The kzorp kernel module receives and accepts the connections.
- All other functionality is performed by Zorp in the userspace.

For both service types, the kzorp kernel module makes the client-side access control (DAC) decision. Both service types can be configured from a uniform interface using ZMC.

Handling packet filtering in the kernel has the following important consequences:

- Packet filtering rules can match on zones as well, not only on IP addresses.
- Network Address Translation (NAT) is available also in the kernel, therefore it is possible to NAT on packet filtering services. However, not every type of NatPolicy can be used with packet filtering services. For details, see *Section 6.7.5, NAT policies (p. 173)*.
- The tproxy table of the iptables utility that earlier Zorp versions used to perform transparent proxying is empty. Zorp does not use it, however it is available if manual adding of rules is necessary.

2.2.4. Proxying connections

Zorp is a proxy gateway. It separates the connection between the client and the server into two separate connections: one between the client and Zorp, and another between Zorp and the server. Zorp receives the incoming client connection requests, inspects them, and transfers them to the server. Zorp also receives the replies of the server, inspects them, and replies to the client instead of the server. That way Zorp has access to the entire network communication between the client and the server, and can enforce protocol standards and the security policy of the organization (for example, permit only specific clients to access the server, or enforce the use of strong encryption algorithms in the connection).

Proxying can take two basic forms:

- **Non-transparent**: In case of non-transparent proxying, client connections target Zorp instead of their intended destination.This solution usually requires some client-side setup, for example, to configure the proxy settings in the web browser of the client.
- **Transparent**: Zorp can operate transparently, to be integrated to the network easily. In case of transparent proxying, the client connections target the intended destinations server, and Zorp inspects the network traffic directly. The client and the server do not detect that Zorp mitigates their communication. In case of transparent proxying, no client side setup is required. Consequently, there is no need to modify the client and server configuration when Zorp is integrated into the network: Zorp is invisible for the end user.

2.2.5. Traffic analysis with proxies

The traffic in a connection usually consists of two parts:

- control information (for example, headers and metainformation)
- data (the payload)

The protocol proxies of Zorp analyze and filter the control part of the traffic, but — in most cases — ignore the payload. (The antivirus and spam-filtering modules of ZCV inspect the payload.) Zorp proxies can thoroughly inspect the protocol headers to ensure compliance to the protocol, disable the use of prohibited options, and so on. Zorp can handle commonly used protocols, including:

- FTP/FTPS
- HTTP/HTTPS
- IMAP/IMAPS
- NNTP/NNTPS
- POP3/POP3S
- RDP
- SIP
- SMTP/SMTPS
- SQLNet
- SSH
- SSL/TLS
- Telnet
- VNC

Every protocol proxy can handle the SSL/TLS encrypted version of the protocol, and inspect the embedded traffic, giving control over HTTPS, SMTPS, and other connections.

For more information on supported protocols and for a complete list of proxies, see Zorp Professional 7 Reference Guide.

2.2.6. Proxy customization

The default settings of the protocol proxies of Zorp ensure that the traffic complies with the relevant RFC of the given protocol. To provide flexibility, and to solve a wide variety of custom scenarios, the proxies can be customized and their parameters can be changed to best suit the needs of the environment and the security requirements. For example, it is possible to:

- disable selected commands in FTP
- modify the transferred headers in HTTP
- permit using only selected web browsers
- specify which encryption algorithms are permitted in SSL/TLS

In addition, the proxies in Zorp are fully scriptable, and can be programmed in Python to perform any custom functionality. For information on customizing proxies, see *Section 6.6.1, Customizing proxies (p. 148)*, and Zorp Professional 7 Reference Guide.

2.2.7. Modular architecture

Today, network traffic often uses more than a single protocol: it embeds another protocol into a transport protocol. For example, HTTPS is HTTP protocol embedded into the Secure Socket Layer (SSL) protocol. SSL encrypts HTTP traffic and many firewalls simply permit encrypted traffic pass without thorough inspection. This is not an optimal solution from a security aspect, and Zorp has a better solution to this problem: it decrypts and inspects the SSL traffic, and passes the data stream to an HTTP proxy to inspect it. This modular architecture (that is, proxies can be stacked into each other, or even chained together for sequential protocol analysis) allows for sophisticated inspection of complex traffic, for example, to perform virus filtering in HTTPS, or spam filtering in POP3S traffic. The new and enhanced version of the integrated category-based URL filtering solution is available from Zorp 7.0.5 with the smaller-sized, *optimized database* for usual scenarios, which requires 1 GB storage space and 300 MB daily update traffic, and with the more extensive *normal database* for more extended scenarios, which needs 6 GB storage space and 2 GB daily update traffic.

Chapter 3. Managing Zorp hosts

There are two ways to manage Zorp firewalls:

- using the Zorp Management Server (ZMS) and the Zorp Management Console (ZMC) graphical user interface (GUI)
  Zorp and ZMS are designed to be configured using the Zorp Management Console (ZMC). The *Zorp Professional 7 Administrator Guide (p. i)* focuses primarily on the preferred ZMC-based administration method.

- manually editing the configuration files of every host
  This method requires advanced Linux skills and deep technical knowledge about how Zorp works. For more information on this procedure, see *Chapter 10, Local firewall administration (p. 235)*.

Note
The two administration methods cannot be mixed: once the editing of the configuration files is started manually, it cannot be continued or be reverted to using ZMC, unless the configuration is rebuilt from scratch.

3.1. ZMS and ZMC

ZMC itself is just a graphical frontend to ZMS. It is ZMS that stores configuration information and manages connections with the agents on managed hosts. The ZMS-based firewall management can only be performed through ZMC; there is no console alternative. ZMC is not “bound” to a particular ZMS host, as long as the proper username/password pair is known, it can be used to connect to any ZMS host.

ZMC can be started the following way:

- *Windows*: Locate the Zorp folder in the **Start** menu and click on the **ZMC** icon. If no such folder has been created when ZMC was installed, start `zmc.exe` manually from the installation folder.
- *Linux*: Start ZMC from the **Network** or **Internet** menu of the desktop environment, or from the console by executing the following command: `./<installation-directory>/bin/zmc`.

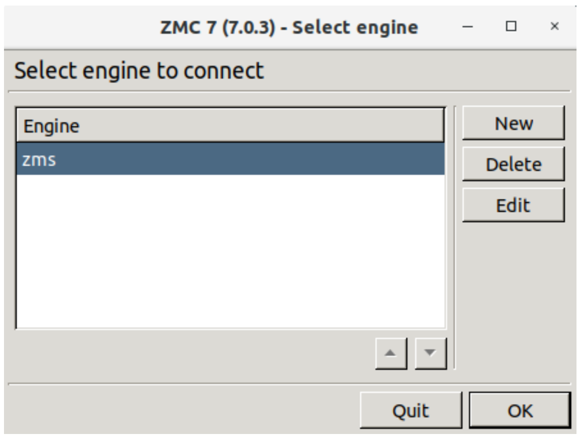


Figure 3.1. Selecting the ZMS engine to connect to

When ZMC is first started after the installation, the list of reachable ZMS hosts is empty, therefore a new host must be defined. To define a new host, click **New**. ZMC configurations are stored in a folder named `.zmsgui`. This folder (for the Windows version of ZMC) is created in the installing user's profile directory, which is typically `%systemroot%\Documents and Settings\%username%`. The name of the file that stores ZMC configurations is `zmsgui.conf`. The Linux version of ZMC stores configuration information in the same manner, within the user's home directory.

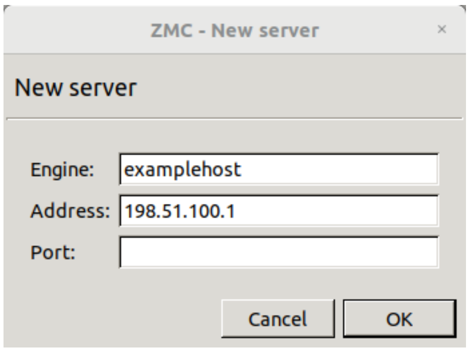
3.1.1. Procedure – Defining a new host and starting ZMC

Purpose:

To define a new host entry and start ZMC, complete the following steps.

Steps:

Step 1. To define a new entry in the list of reachable ZMS hosts, click **New**.



Engine: examplehost
Address: 198.51.100.1
Port:

Figure 3.2. Defining a new host in ZMC

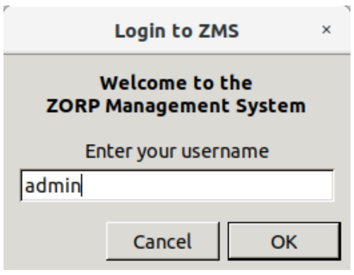
Step 2. Enter a name for the host in the **Engine** field. It can be an arbitrary string and does not have to be the same as the hostname of the ZMS Host.

Step 3. Enter the IP address of the host in the **IP address** field.

Step 4. *Optional step:* Fill the **Port** field or leave it empty to use the default TCP port `1314`. The post assignment can be changed later, if needed.

Step 5. Click **OK**. The new entry is now listed in the **Engine** list.

Step 6. To continue with authentication, click **OK**.



Welcome to the
ZORP Management System

Enter your username

admin

Figure 3.3. ZMC authentication

By default, the built-in administrator account of ZMS and therefore Zorp is `admin`. Its ZMS password was defined during installation.

The name of the administrator can be changed or additional administrators can be added later through ZMC. To modify existing users or add new ones, see *Section 13.1.1, Configuring user authentication and privileges (p. 325)*. To create user accounts with limited privileges (for example, users who can only look at the configuration for auditing purposes, but cannot change anything) see *Section 13.1.1.4, Configuring user privileges in ZMS (p. 328)*.

Expected outcome:

After entering the correct password, if network connectivity is provided, a greeting appears on the ZMC main screen.

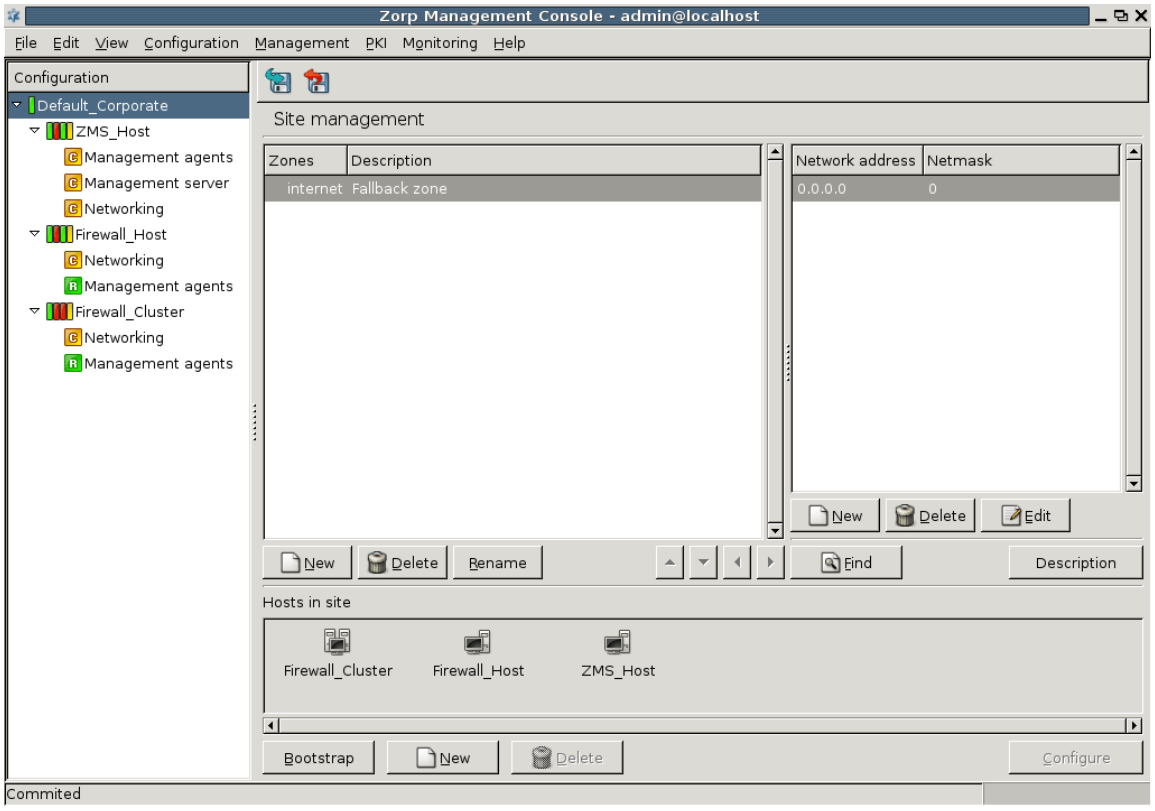
Note
When ZMC connects to a ZMS engine for the first time, it displays the SSH-style fingerprint of the ZMS host. During later connections, it checks the fingerprint automatically.

Warning
ZMC and the ZMS to be accessed must have matching version numbers, that is, ZMS 7.0.1 must be accessed with ZMC 7.0.1. Login is not permitted if the version number of ZMS and ZMC is different.

3.2. ZMC structure

ZMC is divided into three main parts, as presented in the figure:

- *Section 3.2.1, Configuration tree (p. 18)*
- *Section 3.2.2, Main workspace (p. 23)*
- *Section 3.2.3, Menu & status bars and Preferences (p. 24)*



Zones	Description
internet	Fallback zone

Network address	Netmask
0.0.0.0	0

Figure 3.4. ZMC main screen

Note
For more information on the configuration buttons of the button bar, see *Section 3.3.2, Configuration buttons (p. 32)*.

The following sections introduce ZMC components and highlight their main purposes.

3.2.1. Configuration tree

The configuration tree lists the configurable components of a Zorp system. Whenever an item is selected in the configuration tree, the main workplace displays the configurable parameters of the selected item. The configuration tree is organized hierarchically and this hierarchy maps the management philosophy of Zorp.

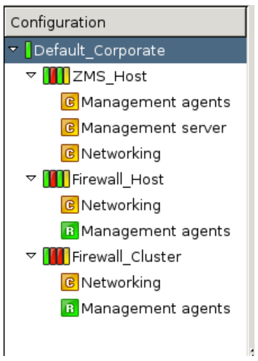


Figure 3.5. Configuration tree in ZMC

The topmost item in the configuration tree is the **Site**'s name that has been entered during ZMS host installation. There are usually one or more items below it: ZMS and/or Zorp hosts.

In the most basic scenario, where ZMS is installed on the Zorp machine, there is only one machine listed. Note that in this case the name that appears here is the name of the ZMS host entered during installation. Under each host, a varying number of configuration components are listed.

By default two components are available for each host:

- `Management agents`
- `Networking`

Because the ZMS Host in this example is a `Management server` too, it has a third component for configuring management server parameters.

Each site, host, and component has status icons or leds on its left. These are described in detail in *Section 3.3.6, Status indicator icons (p. 42)*.

The number of components increases with the configuration: many services have standalone configuration components that that have to be added to the configuration tree to be able to use them. The number of components increases as you start the real work: many services have standalone configuration components that you have to add to the configuration tree to use them.

The forthcoming chapters deal with these components in detail.

3.2.1.1. Site

The biggest configuration entity most Zorp systems consist of is the **Site**. A **Site** is a collection of network entities that belong together from a networking aspect.

From the firewall administration point of view, the **Site** is the collection of the machine nodes. If the company is large and/or has geographically separated subdivisions, more than one firewall may be required. If they are all administered by a single (team of) administrator(s), they can all fall under the supervision of a single ZMS host. In this case, the **Site** consists of a ZMS Host and a number of firewalls.

The reverse of this setup is not possible: a single Zorp firewall cannot be managed by more than one ZMS host, because this setup would cause indefinite and confused firewall states.

If the High Availability (HA) module is also purchased for Zorp and therefore there are two firewall nodes clustered, they can be administered as a single ZMS host. Clusters are described in detail in *Chapter 12, Clusters and high availability (p. 288)*.

ZMC machines do not belong to the **Site**(s) they administer technically, though physically they are located in close proximity to them.

A **Site** is a typical container unit and the components of a **Site** (that is, the **Host**s) share only a few but important properties:

- Zone configuration
  All **Host**s (firewalls) belonging to the same **Site** share a common zone configuration. For more information on zones, see *Chapter 6, Managing network traffic with Zorp (p. 87)*.

- Public key infrastructure (PKI) settings
  Zorp makes heavy use of PKI, for example, in securing communication between ZMS and the firewalls, in authenticating IPSec VPN tunnels, proxying SSL-encrypted traffic.

Although a **Site** can be managed by a single ZMS **Host** only, a ZMS **Host** can manage more than one site.

Tip
A possible reason for a company to create more than one site may be to maintain different Zone structures for different sets of firewalls. This is a frequent requirement for geographically distributed corporations that have separated network segments defended by Zorp firewalls, but want to maintain central (ZMS-based) control over their firewalls.

Another possible user of multi-site, single-ZMS setups is a support company that performs outsourced Zorp administration for a number of clients. In this scenario all business clients are ordered into separate sites, but all these sites are managed by the support company's single ZMS **Host**.

3.2.1.2. Host

A **Site** is composed of one or more **Host**s. **Host**s can be the following items:

- Zorp firewalls,
- ZCV hosts,
- ZAS hosts, and
- ZMS-managed hosts.

At the very minimum setup, when Zorp and ZMS are installed on the same machine, there is one **Host** registered for the given **Site**. The number of Zorp firewall nodes per **Site** is only limited by the number of licenses purchased. With the exception of Zone, PKI and Class Editor settings, the configuration parameters are always per **Host**.

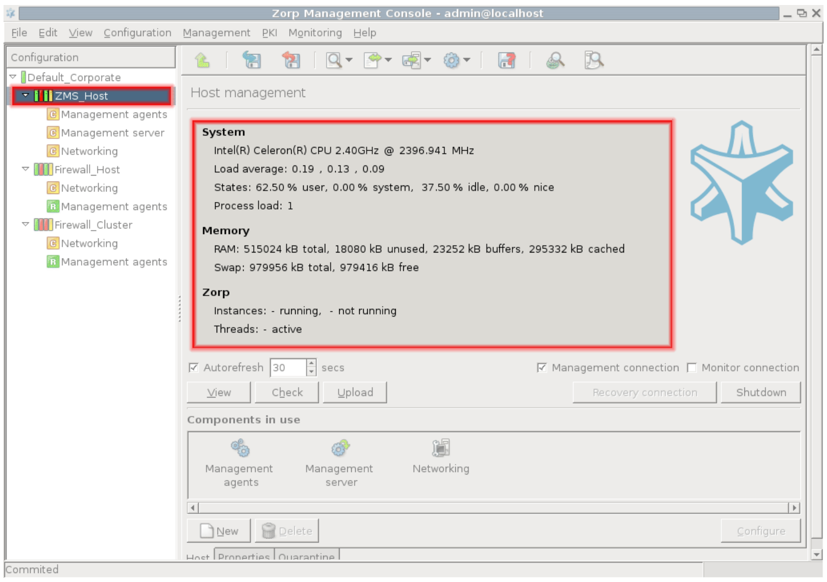
To display system statistics for a **Host** component (ZMS or Zorp), click on the name of the **Host**. The statistics are displayed under the **Host** tab. The following statistics are available:

- **processor**
- **load average**
- **uptime**

- **memory** usage: **RAM** and **SWAP**
- only on Zorp hosts:
  - the version number of Zorp
  - the number and status of running Zorp **Instances**, **Processes** and **Threads**
- the validity and size of the product licenses (Zorp, ZMS, and so on) available on the host. ZMC displays a warning if a license expires soon, and an alert e-mail can be configured as well. For details on configuring e-mail alerts for license expiry, see *Procedure 11.3.8.8, Monitoring licenses and certificates (p. 286)*.

Note
To access license information from the command line, login to the host and enter:

```
/usr/lib/zms-transfer-agent/zms_program_status hoststat
```



```
System
  Intel(R) Celeron(R) CPU 2.40GHz @ 2396.941 MHz
  Load average: 0.19 , 0.13 , 0.09
  States: 62.50 % user, 0.00 % system, 37.50 % idle, 0.00 % nice
  Process load: 1
Memory
  RAM: 515024 kB total, 18080 kB unused, 23252 kB buffers, 295332 kB cached
  Swap: 979956 kB total, 979416 kB free
Zorp
  Instances: - running, - not running
  Threads: - active
```

Figure 3.6. Host statistics

The statistics are automatically refreshed every 30 seconds by default.

Tip
Although host statistics can seem a less important, auxiliary service, it is extremely useful when firewalls operate under continuous heavy load and resource allocation needs to be optimized.

3.2.1.3. Component

The actual configuration of hosts is performed using configuration components. These components are bound to the specific firewall services. For example, there are separate components for Postfix (**Mail transport**), for

NTP (**Time**) and for Zorp itself. The list of usable components depend on the type of host under configuration. Most components belong to Zorp firewall hosts.

By default, there are two components added for each host: `Networking` and `Management Agent`. For ZMS hosts the `Management Server` component is added automatically.

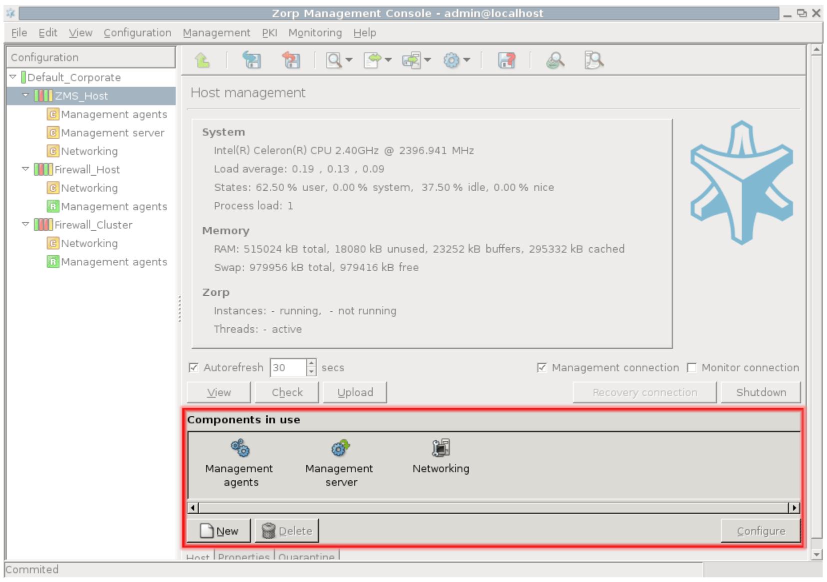
3.2.1.3.1. Procedure – Adding new configuration components to host

Purpose:

To add a new confiugration component to a host, complete the following steps.

Steps:

Step 1. Select the host in the **Configuration** tree, the new component is required to be added to.



System
Intel(R) Celeron(R) CPU 2.40GHz @ 2396.941 MHz
Load average: 0.19 , 0.13 , 0.09
States: 62.50 % user, 0.00 % system, 37.50 % idle, 0.00 % nice
Process load: 1

Memory
RAM: 515024 kB total, 18080 kB unused, 23252 kB buffers, 295332 kB cached
Swap: 979956 kB total, 979416 kB free

Zorp
Instances: - running, - not running
Threads: - active

[✓] Autorefresh 30 secs [✓] Management connection [ ] Monitor connection

Figure 3.7. Adding new components

Step 2. Navigate to the **Host** tab, and under the **Components in use** section, click **New**.

Step 3. Select the configuration component to add from the **Components available** list.

Note
For managing Zorp firewall hosts, it is essential to add the `Zorp` and the `Packet Filter` components, at the very minimum.

The configuration components are strictly focused on the service they manage and all have a distinctive graphical management interface accordingly. For more information on the different components, see the respective chapters.

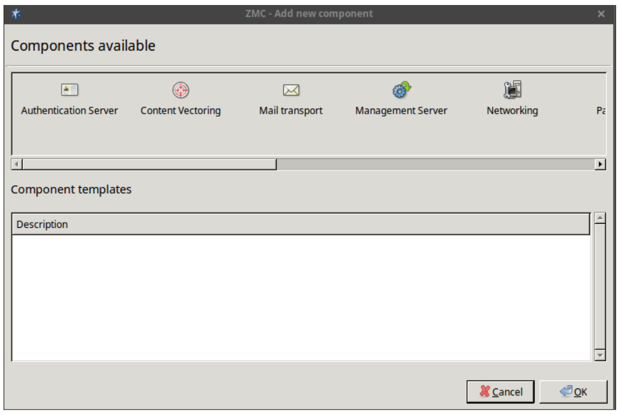


Figure 3.8. Configuration components

The following components are available:

- **Authentication Server**: Zorp Authentication Server (ZAS)
- **Content Vectoring**: Zorp Content Vectoring System (ZCV)
- **Mail transport**: POSTFIX
- **Transfer Agent**:
- **Management Server**: Zorp Management Server (ZMS)
- **Networking**:
- **Packet filter**:
- **System logging**: syslog-ng
- **Text editor**:
- **Time**: NTP
- **VPN**:
- **Zorp**:

Step 4. Select the template to use for the component from the **Component templates** list.

Step 5. Depending on the component, either enter default values for the component in the appearing new window or select a default configuration template.
These built-in templates are configuration skeletons with some default values and options preset. Creating new configuration templates is also possible.

Step 6. Click **OK**.

3.2.2. Main workspace

Most of ZMC is occupied by the main workspace where the variuos components of the **Configuration** tree can be managed. The majority of configuration tasks are performed here. The content of this pane depends on which component is selected in the configuration tree.

The new administrative components can be added to the host at the bottom part of the main workspace.

If a different **Host** is selected in the **Configuration** tree, the content of the main workspace changes too.

Tip
The keyboard shortcuts of ZMC are listed in *Appendix B, Keyboard shortcuts in Zorp Management Console (p. 483)*.

3.2.3. Menu & status bars and Preferences

Although most options of the menu bar are available as buttons on other parts of the ZMC window, there are some special menu points that have no corresponding button in the main workspace. The buttons are described in the specific section which deals with the corresponding ZMC part they appear in.

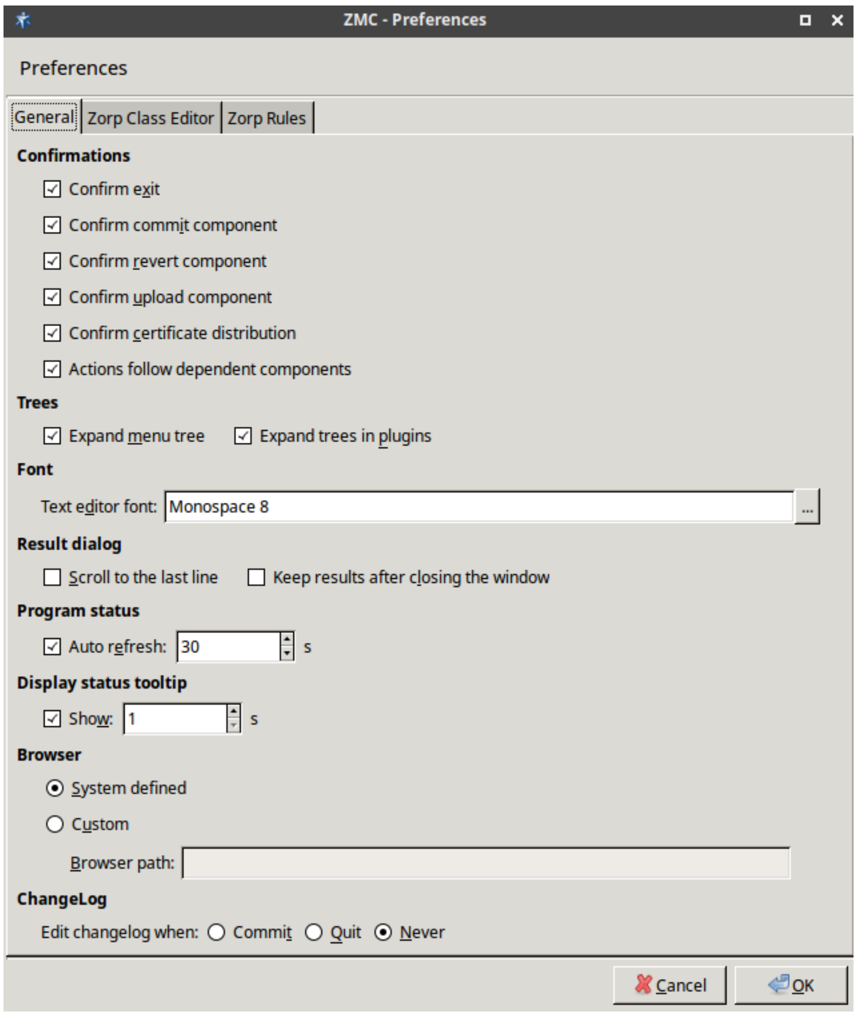
3.2.3.1. Procedure – Configuring general ZMC preferences

Purpose:

To configure general ZMC preferences, complete the following steps.

Steps:

Step 1. Navigate to **Edit > Preferences...** and select the **General** tab.


ZMC - Preferences

Preferences

General | Zorp Class Editor | Zorp Rules

Confirmations

☑ Confirm exit

☑ Confirm commit component

☑ Confirm revert component

☑ Confirm upload component

☑ Confirm certificate distribution

☑ Actions follow dependent components

Trees

☑ Expand menu tree ☑ Expand trees in plugins

Font

Text editor font: Monospace 8

Result dialog

☐ Scroll to the last line ☐ Keep results after closing the window

Program status

☑ Auto refresh: 30 s

Display status tooltip

☑ Show: 1 s

Browser

◉ System defined

○ Custom

Browser path:

ChangeLog

Edit changelog when: ○ Commit ○ Quit ◉ Never

*Figure 3.9. **Edit > Preferences... > General** - Editing ZMC preferences*

Step 2. Edit confirmation window preferences in the **Confirmations** section:

- To display a confirmation window before quitting ZMC through **File** > **Quit** or `Ctrl+Q`, select **Confirm exit**.
- To display a confirmation window before committing the configuration changes to a component, select **Confirm commit component**.
- To display a confirmation window before reverting the configuration changes to a component, select **Confirm revert component**.
- To display a confirmation window before uploading the configuration changes to a component, select **Confirm upload component**.
- **Commit** and **Upload** can be combined into a single action, so that if the configuration changes are required to reach the firewall immediately -and not just the ZMS database-, it can be done with a single click. To combine **Commit** and **Upload**, select **Actions follow dependent components**.

Step 3. Edit tree-related preferences in the **Trees** section:

- **Expand menu tree**.
- **Expand trees in plugins**.

Step 4. Edit text editor font-specific preferences in the **Font** section:

To configure the **Text editor font** of ZMC, click ... . Select the font **Family**, **Style** and the font **Size** and click **OK**.

Step 5. Edit result dialog-specific preferences in the **Result dialog** section:

- **Scroll to the last line**.
- **Keep the results after closing the window**.

Step 6. The status of the interfaces is automatically updated periodically. To configure the update frequency, edit the **Program status** section:
Auto refresh in seconds.

To turn off auto-refresh, deselect **Auto refresh**.

Step 7. Edit the length of the delay after status tooltips are displayed when hovering the mouse over the status led or status icon in the **Display status tooltip** section:
Enter the value of the delay in **Show** in seconds.

To turn off status tooltips, deselect **Show**.

For details on status, see *Section 3.3.6, Status indicator icons (p. 42)*.

Step 8. Edit browser-specific preferences in the **Browser** section:
The *Zorp Professional 7 Administrator Guide (p. i)* and *Zorp Professional 7 Reference Guide* are automatically installed with ZMC in HTML format and are accessible from the **Help** menu. The guides are opened in the default **System defined** browser.

To configure a different browser, select **Custom** and enter in the **Browser path** field the full path name of the browser to use.

Tip
The latest version of these guides, as well as additional whitepapers and tutorials are available at the *Balasys Documentation Page*.

Step 9. Edit changelog-specific preferences in the **ChangeLog** section:
ZMS now records the history of configuration changes into a log file. The logs include who and when modified which component of the Zorp Gateway system. Component restarts and other similar activities are also logged, and the administrators can add comments to every action to make auditing easier. By default, ZMC displays a dialog automatically to comment the changes every time the ZMS configuration is modified, or a component is stopped, started, or restarted. The changelogs cannot be modified later.

For details on writing changelog comments, see *Procedure 3.3.4, Recording and commenting configuration changes (p. 39)*.

To configure when ZMS automatically opens the **New changelog entry** window, change **Edit changelog when** to:

- **Commit** to automatically open the **New changelog entry** window after committing the configuration changes to a component.
- **Quit** to automatically open the **New changelog entry** window only before quitting ZMC through **File** > **Quit** or `Ctrl+Q`.
- **Never** if automatical open of the **New changelog entry** window is never required.

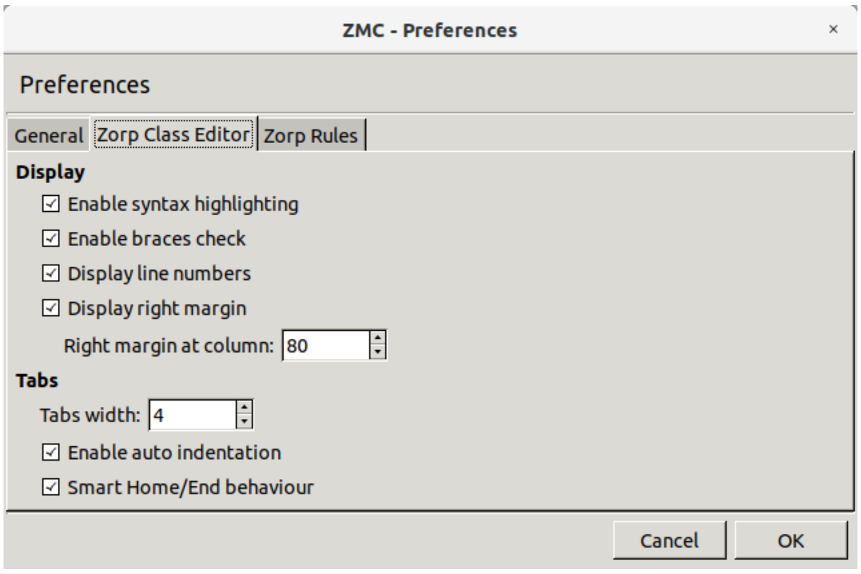
3.2.3.2. Procedure – Configuring Zorp Class Editor preferences

Purpose:

To configure Zorp Class Editor preferences, complete the following steps.

Steps:

Step 1. Navigate to **Edit** > **Preferences...** and select the **Zorp Class Editor** tab.



Display
- ☑ Enable syntax highlighting
- ☑ Enable braces check
- ☑ Display line numbers
- ☑ Display right margin

Right margin at column: 80

Tabs

Tabs width: 4
- ☑ Enable auto indentation
- ☑ Smart Home/End behaviour

*Figure 3.10. **Edit > Preferences... > Zorp Class Editor** - Editing Zorp Class Editor preferences*

Step 2. Configure display-related preferences in the **Display** section:

- To enable syntax highlighting, select **Enable syntax highlighting**.
- To enable syntax validation by checking braces, select **Enable braces check**.
- To display line numbers, select **Display line numbers**.
- To display right margin, select **Display right margin**. Enter the column number where the margin line is required to be displayed in **Right margin at column**.

Step 3. Configure tabulation in the **Tabs** section:

- Enter the indentation width in spaces in the **Tabs width** field.

- For auto-indentation, select **Enable auto indentation**.
- For smart behavior for **Home** and **End** keys, select **Smart Home/End behaviour**.

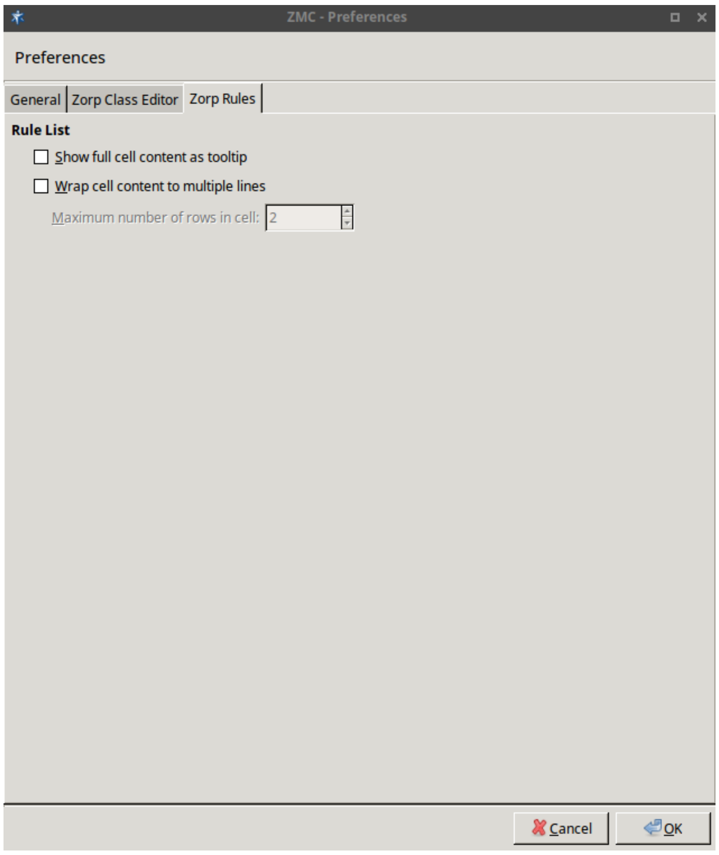
3.2.3.3. Procedure – Configuring Zorp Rules preferences

Purpose:

To configure Zorp Rules preferences, complete the following steps.

Steps:

Step 1. Navigate to **Edit** > **Preferences...** and select the **Zorp Rules** tab.



Rule List

☐ Show full cell content as tooltip

☐ Wrap cell content to multiple lines

Maximum number of rows in cell: 2

Figure 3.11. Editing ZMC preferences

Step 2. Configure rule list-related preferences in the **Rule list** section:

- To display the complete content of a cell as tooltip, select **Show full cell content as tooltip**.
- To wrap cell content if it exceeds a certain length, select **Wrap cell content to multiple lines**. Define the **Maximum number of rows in cell**.

3.2.3.4. Procedure – Configuring ZMS hosts

Purpose:

To add, delete or edit ZMS hosts, complete the following steps.

Steps:

Step 1. Navigate to **Edit** > **Servers...**.

Step 2.

- To add a new host, click **New**. For details on the configuration steps, see *Procedure 3.1.1, Defining a new host and starting ZMC (p. 16)*.

  Note
  ZMC cannot connect to more than one ZMS host simultaneously. After adding a new host, ZMC will not change to that new host, but will stay logged in to the host that is currently being configured. To configure the new host, navigate to **File** > **Relogin...** and login to the new host.

- To edit an already existing host, click on the name of the host and click **Edit**.
- To delete an already existing host, select the name of the host and click **Delete**.

  Warning
  There is no confirmation window after clicking **Delete**, the host is deleted instantly. Make sure that **Delete** is used only if a host is required to be deleted.

Step 3. Click **Close**.

3.2.3.5. PKI menu

PKI settings are always site-wide and can be configured graphically using the PKI menu only. For more information on PKI usage under Zorp, see *Chapter 11, Key and certificate management in Zorp (p. 249)*.

3.2.3.6. Variables menu

To clarify management, it is possible to define system variables for ZMS. These variables all have a scope, depending on which component is selected in the **Configuration** tree when they are declared.

Altogether, variables can work in three scopes that correspond to configuration levels in the **Configuration** tree: **Site**, **Host** and **Component**.

Tip
Using variables is especially useful in sophisticated, enterprise Zorp environments where complex configurations have to be maintained. When referencing variables inside configuration windows, $ characters must precede and follow their names. For example, `$autobind-ip$`.

By using variables it is simpler and error-free to change a value that is present at many different places. Modifying a corresponding variable results in changed values everywhere they are used.

Note
Instead of variables, it is recommended to use links.

By default there are two host variables defined:

- Hostname, and
- `autobind-ip`.

3.2.3.6.1. Procedure – Defining variables

Steps:

Step 1. Select a **Site**, **Host** or **Component** in the configuration tree.

Step 2. Navigate to **Edit** > **Variables...**.

Step 3. To create a new variable, click **New**.

Step 4. Enter the **Name-Value** pair in the respective fields.

Step 5. Click **Close**.

3.2.3.6.2. Procedure – Editing variables

Steps:

Step 1. Select a **Host** in the configuration tree.

Step 2. Navigate to **Edit** > **Variables...**.

Step 3. To edit a variable, select the variable and click **Edit**.

Step 4. Enter the new **Name-Value** pair in the respective fields.

Step 5. Click **Close**.

3.2.3.6.3. Procedure – Deleting variables

Steps:

Step 1. Select a **Host** in the configuration tree.

Step 2. Navigate to **Edit** > **Variables...**.

Step 3. To delete a variable, select the variable and click **Delete**.

Warning
There is no confirmation window after clicking **Delete**, the variable is deleted instantly. Make sure that **Delete** is used only if a variable is required to be deleted.

Step 4. Click **Close**.

3.2.3.7. Status bar

The bottom line of ZMC is called the status bar. When working with configuration components it can be used to check whether changes have already been **Committed** or there are **Unsaved changes**. By checking the status, it can be made sure whether the information there on the ZMC interface is the same as the information currently stored in the ZMS configuration database (committed status).

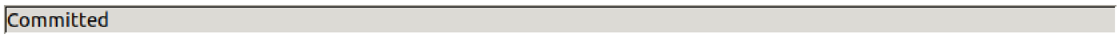


Figure 3.12. The Status bar

3.3. Configuration and Configuration management

Most configuration tasks concerning Zorp are component–based and even those that are site-wide, such as Zone manipulation, must be individually uploaded to all firewalls of the given site. Therefore, configuration tasks can be organized into cycles and most elements of these cycles are the same regardless of the component that is configured. In fact, most of the configuration is repetitive and therefore can easily be procedurized.

In this section, after a brief overview of the most typical steps, the configuration process and the tools (buttons) that are used to perform each task are presented.

3.3.1. Configuration process

When logging in to ZMS through ZMC, first an SSL encrypted channel is built, then firewall configurations currently stored in the ZMS database are downloaded into ZMC. When the required configuration changes are completed they are committed back into the ZMS database. At this point no changes are made to the firewall(s); only the database on the ZMS host is modified. It takes a separate action, an upload issued to actually propagate changes from the database down to the firewall(s). With this upload action the configuration changes get integrated into the configuration files on the Zorp machine(s). For final activation, a reload or restart (depending on the situation and the service being modified) is needed to activate the changes.

A complete configuration cycle consists of the steps described in the forthcoming sub-sections.

3.3.1.1. Procedure – Configuring Zorp - the general process

Step 1. Select the component required to be configured in the **Configuration** tree.

Step 2. Perform the actual configuration changes on the component. For details, see the relevant chapters.

Step 3. Commit the changes to the XML database of ZMS. Otherwise, the changes are lost when navigating to another component.
Write a brief summary about the changes into the **Changelog**. For details, see *Procedure 3.3.4, Recording and commenting configuration changes (p. 39)*.

Step 4. To activate the changes, upload them to the affected Zorp firewall hosts from the ZMS database. ZMS converts the changes to the proper configuration file format and sends them to the transfer agents on the firewall nodes. The changes are applied on the firewall nodes.

Step 5. Reload the altered configurations on the firewalls, or restart the corresponding services.

Note
Not all of these steps are performed in each configuration cycle. Service reloads or restarts are typically postponed as long as possible and are likely to be performed only after all configuration tasks with the various service components are finished.

3.3.2. Configuration buttons

Most administration commands for the configuration tasks can be executed from either the menus or the buttons in the Button bar. The number of buttons visible varies based on the component selected in the **Configuration** tree.

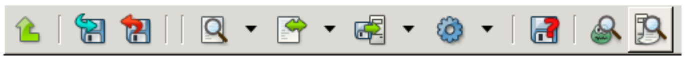


Figure 3.13. The Button bar

3.3.2.1. Commit and Revert

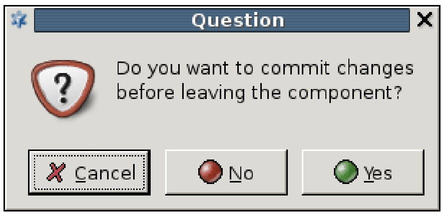
The **Commit changes** and **Revert changes** buttons are always visible, at the minimum.

Commit is used when a (set of) configuration changes are finished and the changes are required to be saved to the XML database of ZMS.

Revert serves the opposite purpose: before committing changes to the ZMS database, it is possible to clear, to undo them in ZMC.

Note
It is very important to remember that **Revert** is limited to ZMC, it cannot clear configuration changes that are already committed to the ZMS database. Those changes can be undone by performing a new round of changes in ZMC and then committing these changes again.

Both **Commit** and **Revert** are component–focused controls. Consequently, before selecting another component from the **Configuration** tree, commit the changes in the current component, otherwise they are lost. In such cases, ZMC displays the following warning:



Question

Do you want to commit changes before leaving the component?

Figure 3.14. Warning: commit the changes before leaving the component

3.3.2.2. Upload current configuration

Upload is used to upload the configuration changes committed to the configuration database on the ZMS Host further to the corresponding Zorp firewall(s).

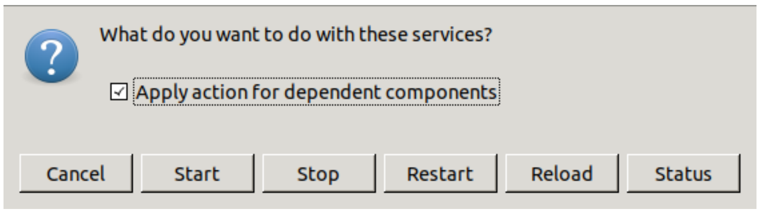
Tip
Commit and **Upload** can be combined into a single action, so that if the configuration changes are required to reach the firewall immediately – and not just the ZMS database –, it can be done with a single click. which means that if you want configuration changes to reach the firewall immediately – and not just the ZMS database – you can do it with a single click. To combine **Commit** and **Upload**, navigate to **View** > **Preferences** and select **Actions follow dependent components**.

3.3.2.3. Control service

Under **Control service**, services can be **Reload**ed or **Restart**ed so that they reread the new configuration files that are already on the corresponding Zorp firewalls after a successful **Commit** / **Upload** cycle.

Restarting or **Reload**ing the given service depends on the type of service (some cannot be reloaded, only restarted) and the intended outcome.

After clicking **Control service**, the following actions are available:


What do you want to do with these services?

☑ Apply action for dependent components

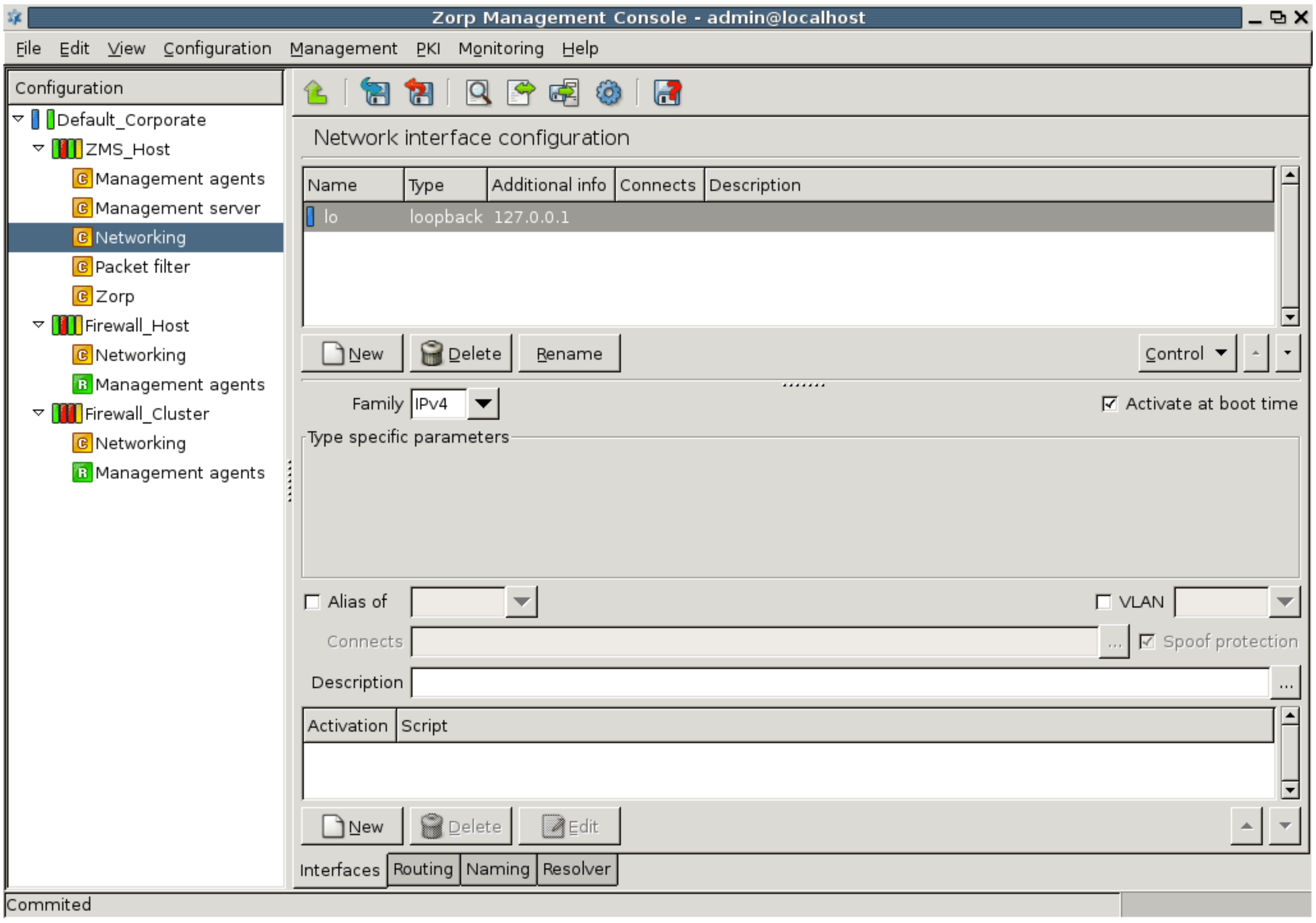
Figure 3.15. The service control dialog

Note
Besides **Restart** and **Reload**, there are also **Start** and **Stop** functions available here to start or stop services.

3.3.2.4. View and Check current configuration

View current configuration and **Check current configuration** are both used to retrieve information on the current state of the Zorp firewall(s).

View current configuration displays the configurations of the component selected in the **Configuration** tree on the selected host. This information comes from the ZMS configuration database, which is not necessarily the same as the actual settings on the selected host – when changes are already committed, but not yet uploaded. For example, if the *`ZMS_Host`* > `Networking` component is selected and then **View current configuration** is used, the following will be displayed:



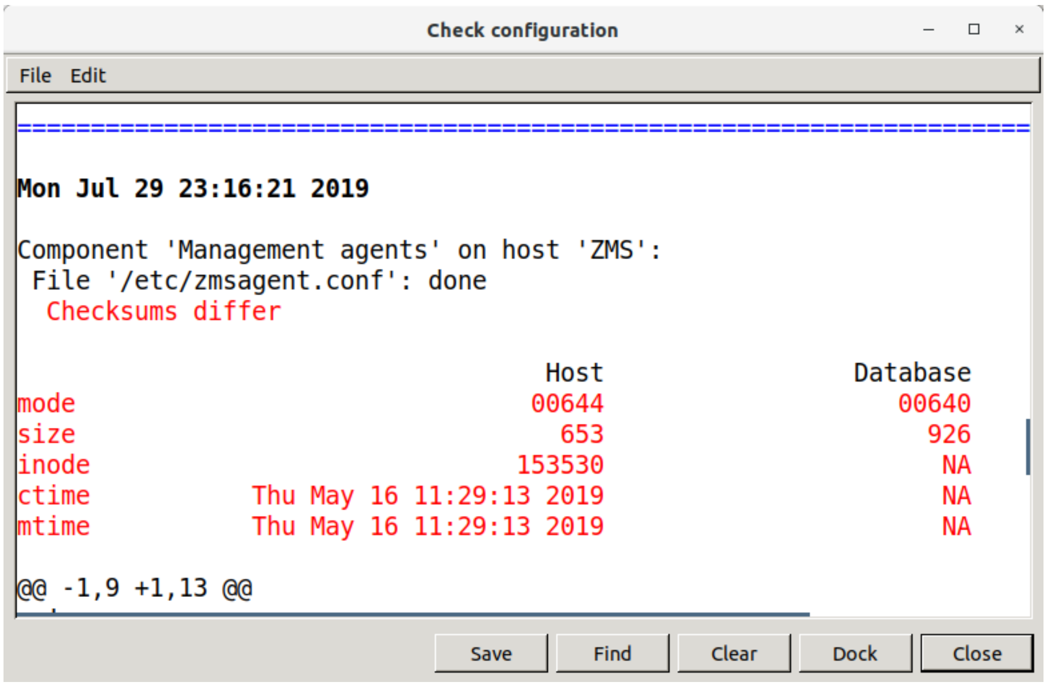
Name	Type	Additional info	Connects	Description
lo	loopback	127.0.0.1		

Family: IPv4

Activation	Script

Figure 3.16. Networking configuration on ZMS_Host

It is a file-by-file listing of the active configuration on the selected host. Note that it is not necessarily the same configuration that is stored in the ZMS database: after a commit but prior to an upload event they can differ significantly. To query this difference, click **Check current configuration**. Using the Linux `diff` utility by default, it compares configurations stored in the ZMS database with the configurations currently active on the selected host.



```
=========================================================================

Mon Jul 29 23:16:21 2019

Component 'Management agents' on host 'ZMS':
 File '/etc/zmsagent.conf': done
  Checksums differ

                                   Host                     Database
mode                              00644                        00640
size                                653                          926
inode                            153530                           NA
ctime                Thu May 16 11:29:13 2019                     NA
mtime                Thu May 16 11:29:13 2019                     NA

@@ -1,9 +1,13 @@
```

Figure 3.17. Checking current configurations

The differences are marked in red, otherwise the normal output of `diff` is displayed, with + and – signs designating data from the host and from the database, respectively. The `diff` command can be replaced with another utility of choice under the `Management Server` component. For details, see *Chapter 13, Advanced ZMS and Agent configuration (p. 323)*.

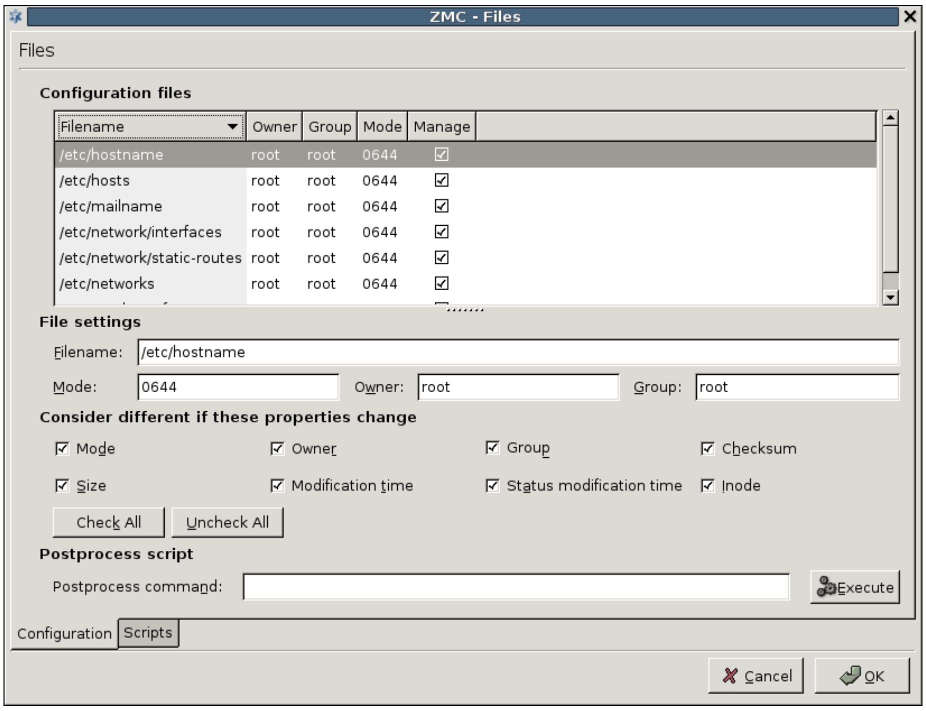
3.3.2.5. Files

Configuration files:

Files provides further information and configuration options of the files and attributes described in the output window of **Check current configuration** and the `diff` command.

Files serves two purposes.: It provides vital information about which configuration files a component (of the **Configuration** tree) uses and gives chance to modify the properties of the listed files.

For example, in case of the `Networking` component, the list of used files is the following.


ZMC - Files

Files

Configuration files

Filename	Owner	Group	Mode	Manage
/etc/hostname	root	root	0644	☑
/etc/hosts	root	root	0644	☑
/etc/mailname	root	root	0644	☑
/etc/network/interfaces	root	root	0644	☑
/etc/network/static-routes	root	root	0644	☑
/etc/networks	root	root	0644	☑

File settings

Filename:	/etc/hostname				
Mode:	0644	Owner:	root	Group:	root

Consider different if these properties change

☑ Mode ☑ Owner ☑ Group ☑ Checksum

☑ Size ☑ Modification time ☑ Status modification time ☑ Inode

Postprocess script

Postprocess command:

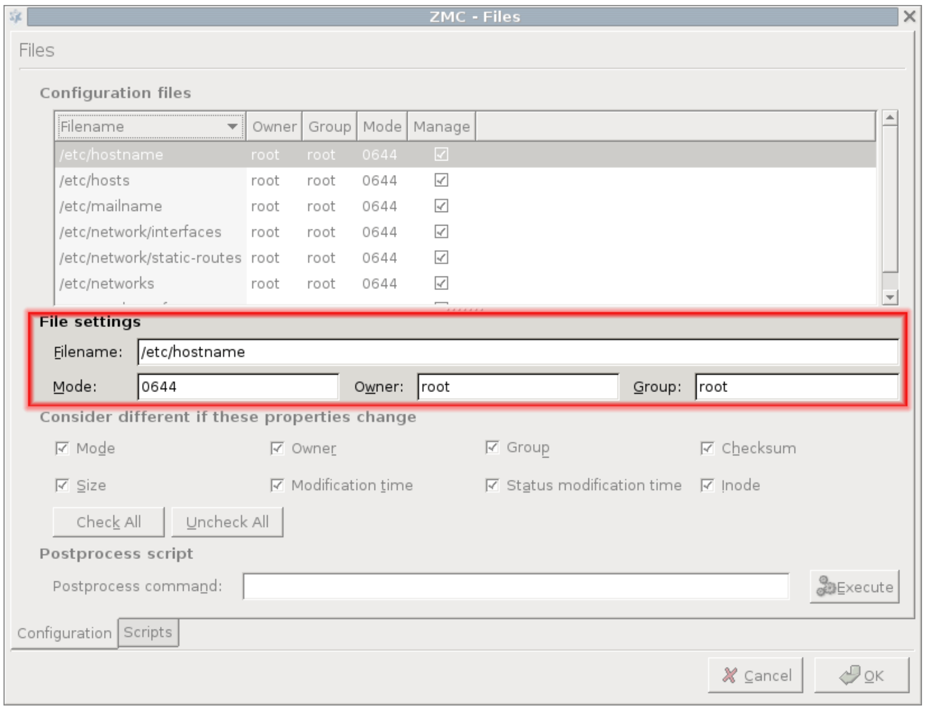
Figure 3.18. Files used by the Networking component

Apart from the name and location of files, information can be retrieved about the owner, owner group, access rights and file type parameters. The **Manage** column is very important and has a corresponding checkbox immediately below the file listing: this can be used to control what files ZMS manipulates on the host machine, if needed.

Note
It is not recommended to take files out of the authority of ZMS, because it can severely limit the effectiveness of ZMS–based administration. However, it is possible to do it, if the checkbox under the **Manage** column is deselected.

File settings:

To modify the properties of a file, click on the file in the list. The following subwindow opens.



Filename	Owner	Group	Mode	Manage
/etc/hostname	root	root	0644	☑
/etc/hosts	root	root	0644	☑
/etc/mailname	root	root	0644	☑
/etc/network/interfaces	root	root	0644	☑
/etc/network/static-routes	root	root	0644	☑
/etc/networks	root	root	0644	☑

File settings

Filename: /etc/hostname

Mode: 0644 Owner: root Group: root

Consider different if these properties change

☑ Mode ☑ Owner ☑ Group ☑ Checksum

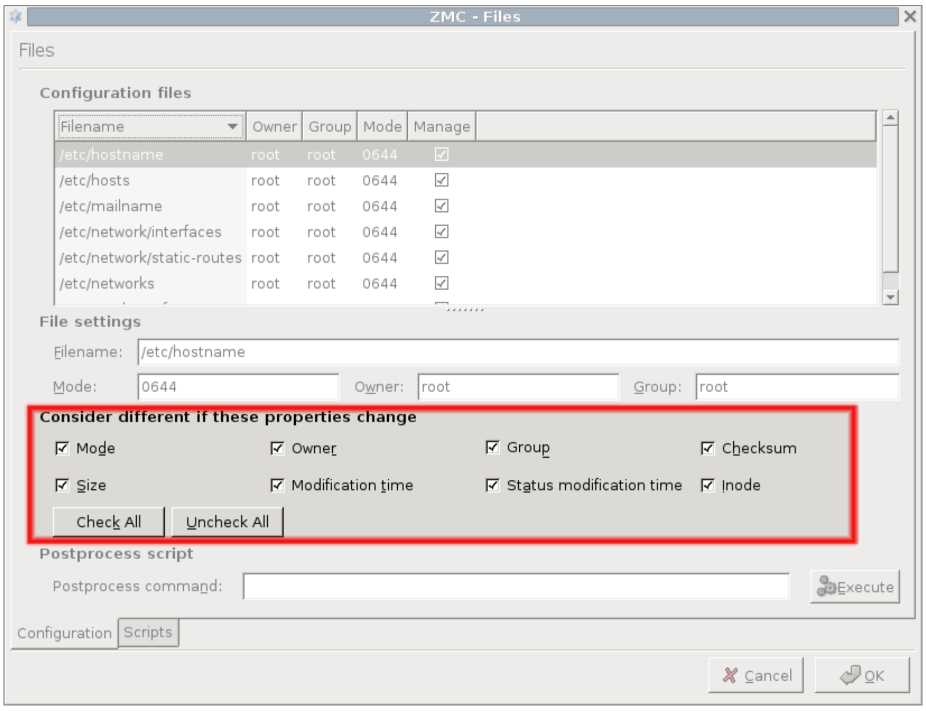
☑ Size ☑ Modification time ☑ Status modification time ☑ Inode

Figure 3.19. Changing file properties

Warning
There must be a solid reason for changing these properties and one must be prepared for the possible consequences of such actions. A good understanding of Linux is recommended before making changes in file properties.

Consider different if these properties change:

The third part of the window is for configuring the work of the comparison utility, which is `diff` by default. It can be defined which file properties are required when checking for changes.


ZMC - Files

Files

Configuration files

Filename	Owner	Group	Mode	Manage
/etc/hostname	root	root	0644	☑
/etc/hosts	root	root	0644	☑
/etc/mailname	root	root	0644	☑
/etc/network/interfaces	root	root	0644	☑
/etc/network/static-routes	root	root	0644	☑
/etc/networks	root	root	0644	☑

File settings

Filename: /etc/hostname

Mode: 0644 Owner: root Group: root

Consider different if these properties change

☑ Mode ☑ Owner ☑ Group ☑ Checksum

☑ Size ☑ Modification time ☑ Status modification time ☑ Inode

Check All Uncheck All

Postprocess script

Postprocess command:

Figure 3.20. Configuring diff conditions

Tip
Checking for configuration file differences is beneficial from a security aspect too: it is an additional tool for making sure nobody has altered critical files on the firewall.

Postprocess script:

At the bottom of the **Configuration** tab, a postprocess command can be specified that is run after the corresponding configuration file is uploaded to the firewall host. Some services rely heavily on this option. For example, Postfix that runs `/usr/sbin/postmap %f` as a postprocess command to transport virtual domains and set various access restrictions are properly.

Scripts tab:

Configuration files under Linux are reread during service reloads or restarts. These actions are performed by running the corresponding scripts exclusively from the `/etc/init.d` directory. The **Scripts** tab of the **Files** window provides an interface where the starting scripts can be checked and alter and fine-tune them with special `Pre upload` and `Post upload` commands. With simple components, such as `Networking`, these options are rarely used, but in some cases might prove especially useful.

Some components, for example, **Text Editor**, can manage configuration files that are automatically reloaded. They cannot be restarted after a **Commit**. To set the status icon of these components to **Running**, select **Configuration automatically runs** on the **Scripts** tab.

3.3.3. Committing related components

Some components are related to or dependent upon each other, meaning that modifying one modifies the other too. If modifying a component affects another component, the status of this related component changes to **Invalidated** in the **Configuration** tree. ZMC automatically handles related components and all actions (**Commit**, **Reload**, and so on). Just select the **Apply action for the dependent components** checkbox in the confirmation dialog of the action.

When reloading or restarting a component related to the Packet Filter, the skeleton of the Packet Filter is automatically regenerated. See *Appendix A, Packet Filtering (p. 462)* for details on generating skeletons.

3.3.4. Procedure – Recording and commenting configuration changes

Purpose:

ZMS now records the history of configuration changes into a log file. The logs include who and when modified which component of the Zorp Gateway system. Component restarts and other similar activities are also logged, and the administrators can add comments to every action to make auditing easier. By default, ZMC displays a dialog automatically to comment the changes every time the ZMS configuration is modified, or a component is stopped, started, or restarted. The changelogs cannot be modified later.

The behavior of the changelog window can be configured in **Edit** > **Preferences** > **General**. For details, see *Procedure 3.2.3.1, Configuring general ZMC preferences (p. 24)*.

To review the existing changelog entries, navigate to **Management** > **Changelogs**. The window contains two filter bars: the **Changelog entries** filters for changelog entries, the **Components** filters inside a single changelog entry, if it contains too many actions. For details, see *Section 3.3.10, Filtering list entries (p. 48)*.

Steps:

Step 1. *Optional step*: If the **New changelog entry** window is not configured to display automatically after committing or quitting, a new changelog entry can be added manually. To do this, navigate to **Edit** > **Changelog...**.

Step 2. Review the details of the changelog entry. Every changelog entry includes the following information:

- **Date**: It is the date when the action was performed.
- **Administrator**: It provides the ZMC username of the administrator who performed the action.
- **Host**: It is the Zorp **Hosts** affected by the action.
- **Component**: It lists the ZMC components affected by the action.
- **Action**: It is the type of change that was performed.

Step 3. Enter the following:

- **Summary**: Enter a short summary of the changes.
- **Description**: Enter a detailed description of the changes.

Step 4. To save the changelog entry, click **Send**.

3.3.5. Multiple access and lock management

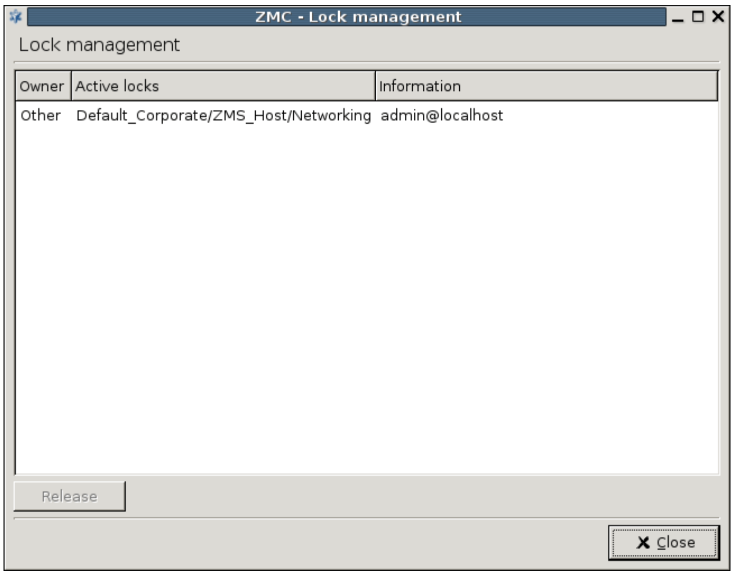
Most firewalls are administered by a group of administrators and not just by a single individual. In a Zorp system each administrator can have his or her own ZMC console and administrators can be separated geographically. Regardless of their locations they administer the same set of Zorp firewalls through a single ZMS host machine. Therefore, to avoid configuration inconsistency caused by more than one administrator working with the same configuration simultaneously, a configuration lock mechanism ensures that a component's configuration can only be modified by a single administrator at a given time. Locking takes place per component, as soon as they are changed, for example, a setting in a component, the status bar displays the following string: **Unsaved changes** and that component is locked for configuration.

However, modifying a component might make it necessary for other components to be locked as well, in order to prevent configuration from inconsistent changes.

There exist some generic rules that can be applied for managing locking successfully:

- A component can only be modified if it is not locked by another administrator.
- Modifying a component results in locking it.
- Reading a component does not imply locking it.
- If a site is modified, and is consequently locked, it implies the locking of PacketFilter, Networking and Zorp components as well on every host of the site.
- If a new component is created, it might need to be locked according to the rules. If for example, another administrator is modifying a component that requires the lock of this newly created component to preserve consistent configuration structure.
- The force of unlock, that is the release of an administrator's lock by someone else, implies shutting down the relevant GUI. Consequently, all the locks will be released, the changes not commited yet will be lost.
  Note that a force unlock has to be consulted with the relevant parties prior to the release of the lock, as it might imply loss of data.
- If any of the changes to a component are reverted, the lock of the actual component is released.
- If changes to a component are committed the lock of the actual component is released.

Active locks can be viewed at **Management** > **Locks**:


ZMC - Lock management

Lock management

Owner	Active locks	Information
Other	Default_Corporate/ZMS_Host/Networking	admin@localhost

*Figure 3.21. **Management > Locks** - Viewing active locks*

The **Owner** column can take two values:

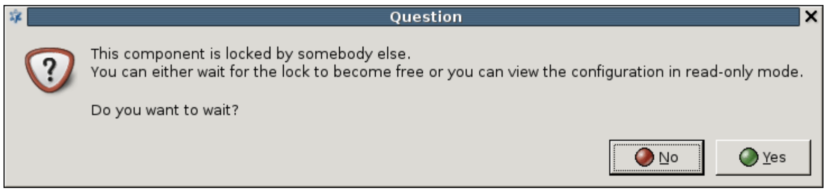
- **Other**
  If someone else is working with the given component.

- **Self**
  Indicating your own lock.

The lock placement is automatic. The first administrator who starts modifying a component's settings gets the lock of that actual component. In the **Active locks** column the exact name of the locked component (**Site**/**Host**/**Component**) is displayed. Locks are cooperative, meaning that any administrator can release any other administrator's locks by selecting the desired component in the **Lock management** window and then clicking **Release**. The administrator whose lock is released this way is immediately notified in a warning dialog. Also note that as a result of the release of the forced unlock, the GUI owning the released lock closes without saving any modifications in order to avoid any inconsistency in the configuration.

Note
As releasing the lock of another administrator is a rather radical interaction, concurrent administrators shall discuss lock situations before possibly devastating each other's work.

It is not possible to edit a component that is already locked by someone else, because a notification dialog immediately appears upon trying to change anything inside the given component. The following window will be displayed for any other administrator who wishes to edit the component:


Question

This component is locked by somebody else.
You can either wait for the lock to become free or you can view the configuration in read-only mode.

Do you want to wait?

Figure 3.22. Notification on a locked component

As required by the mentioned lock queue mechanism implemented in ZMS, administrators shall not normally release each other's locks, but they can preregister for future locks while the current lock is active.

The above window will however not disappear, after the locking administrator commits the changes. Other administrators therefore, can either preregister for locking the component by clicking **yes** or choose not to preregister by clicking **no**.

In case an administrator preregisters for the locking of an element, the following window appears:

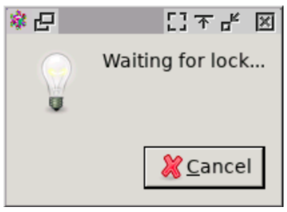

Waiting for lock...

Figure 3.23. Component waiting for locking

As soon as the locking administrator releases the lock, the window *Component waiting for lock* disappears and the user wishing to edit the component is granted the lock automatically. Normally, this shall not take long.

3.3.6. Status indicator icons

The **Configuration** tree in ZMC displays various indicators to provide a quick overview about the status of the managed sites, hosts, and components. **Site** and **Host**-level status is indicated by leds, while icons are used to display **Component**-level status information.

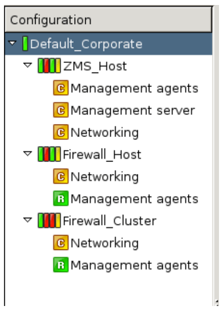
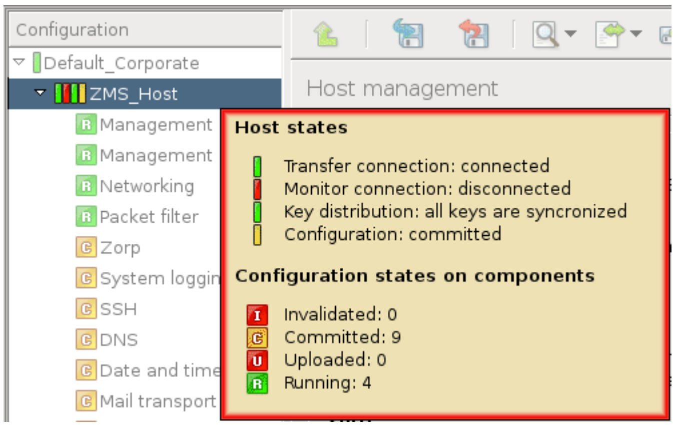


Figure 3.24. Status indicator icons and leds

Hovering the mouse cursor over a led or icon displays a tooltip with the full description of the status.

Tip
Status tooltips are displayed for the period configured in **Edit** > **Preferences** (for details, see *Section 3.2.3, Menu & status bars and Preferences (p. 24)*). It is also possible to disable the tooltips altogether.



Host states

Transfer connection: connected
Monitor connection: disconnected
Key distribution: all keys are syncronized
Configuration: committed

Configuration states on components

Invalidated: 0
Committed: 9
Uploaded: 0
Running: 4

Figure 3.25. Tooltip after hovering the mouse over an icon

Note
Similar leds are also used throughout ZMC to display information about the state of various objects, for example, network interfaces, Zorp instances, NTP servers, Heartbeat resources, and so on. These are described in their respective sections.

3.3.6.1. Site-level indicators

The **Site**-level led displays the validity of the certificates used by the PKI system of the **Site** (for details, see *Chapter 11, Key and certificate management in Zorp (p. 249)*). The led has the following three different states:

- *Green* 🟩: All certificates are valid.
- *Yellow* 🟨: One or more certificate will expire soon.
- *Red* 🟥: One or more certificate has expired.

Note
Expired or soon-to-expire certificates are displayed in bold on in the PKI management tabs.

3.3.6.2. Host and cluster-level indicators

The status of the **Host** is displayed by four different leds. From left to right, these are the following:

- *Transfer and Monitor connection*
- *Key distribution*
- *Configuration*

All four leds can be **Blue**, ▮, indicating a *partial* or unknown status: this appears when the status of the nodes of a cluster differs from each other. For example, the transfer agent led is blue if the agent could establish the connection to only one of the nodes, or if the state of the agent is unknown.

Hovering the mouse pointer above the leds displays a tooltip containing detailed information of the leds, including a summary of the committed/uploaded/and so on components.

Transfer and Monitor connection

These leds indicate the status of the Transfer and Monitoring agent connections, respectively.

- **Green** ▮: the agent is connected to the host.
- **Yellow** ▮: a connection attempt is in progress.
- **Red** ▮: the agent is disconnected.

The management connection leds display `unknown` status if the given connection is not enabled on the host and is in the disconnected state.

Key distribution

This led indicates the availability of the required certificates and keys on the **Host**.

- **Green** ▮: normal state.
- **Red** ▮: the certificates have been modified (for example, refreshed), but the new certificates have not been distributed yet to the **Host**.

Warning
This status led is especially important, because if the certificates are not distributed properly, ZMS will not be able to communicate with the **Host**. For details on distributing certificates, see *Section 11.3.5.2, Distribution of certificates (p. 261)*.

Configuration

The **Component** led indicates only the state of the component that is in the **worst** state. That is, if all components are in normal state, but one of the components is not committed, the led will be in the **unsaved** state.

3.3.6.3. Component-level status indicators

The status of each component on a host (or cluster) is indicated by a single icon. The components can have the following states:

Modified M: The component has been modified, but the changes have not been commited to the ZMS database yet.

Invalidated I: This status indicates that the component has to be updated because of a modification that was performed in another component. For example, committing modifications of the **Zorp** component invalidates the **Packet filter** component, because the packet filtering rules have to be regenerated. Invalidated components automatically become modified when they are selected from the **Configuration** tree.

Committed 🟨: The modifications of the component have been saved to the ZMS database, but the new configuration has not been uploaded to the host. For details, see *Section 3.3.2.1, Commit and Revert (p. 32)*.

Uploaded 🟥: The new configuration has been successfully uploaded to the host. For details, see *Section 3.3.2.2, Upload current configuration (p. 32)*.

Running 🟩: The uploaded configuration has been successfully activated on the host (for example, the service/instance has been restarted/reloaded, and so on). For details, see *Section 3.3.2.3, Control service (p. 33)*.

Partial 🟩 🟥: These states appear when the status of the nodes in a cluster differs from each other. For example, the partial uploaded icon indicates that the new configuration was not successfully uploaded to all nodes.

Locked: The component is in use (and has been modified) by another user (for details, see *Section 3.3.5, Multiple access and lock management (p. 40)*). This status is indicated by the above icons having grayed colors, for example, ⬜.

Occassionally it might be required to manually modify the status of a component. This can be done from the **Configuration** menu through the **Mark as Committed** and **Mark as Running** menu items.

Note
Only uploaded components can be marked as running.

3.3.7. Copy, paste and multiple select in ZMC

ZMC provides two graphical aids that can help administration when parts of a host's configuration settings have to be recreated on another host.

- Copy and paste
  Elements of the configuration (for example, network interfaces, proxies, policies, and so on) can be copied and then pasted to another host. This method can also duplicate an element on the same host. All settings of the element are copied to the target host.

  To copy a configuration element, select the file content for example, right-click and select **Copy**.

  To paste a configuration element, right-click where the element is required to be pasted and select **Paste**.

  Warning
  Make sure to verify the settings of the pasted element, especially the parameters that used links.

- Multiple select
  - To select consecutive multiple components, select the first component, press **Shift** and click the last component.
  - To select multiple components that are not consecutive, select the first component, press **Ctrl** and click the next component for selection.

If multiple components are selected, after right-clicking, the **View**, **Check**, **Upload** and **Control** operations are permitted. After clicking one of the options, all configuration files are batch-processed.

Additionally, a program, for example an archiving script, can be run on the configuration files of all selected components. To do this, either select the command from the drop-down menu or enter the command in the **Run program** field and click **Execute**.

Note
Multiple selected components can also be copy-pasted.

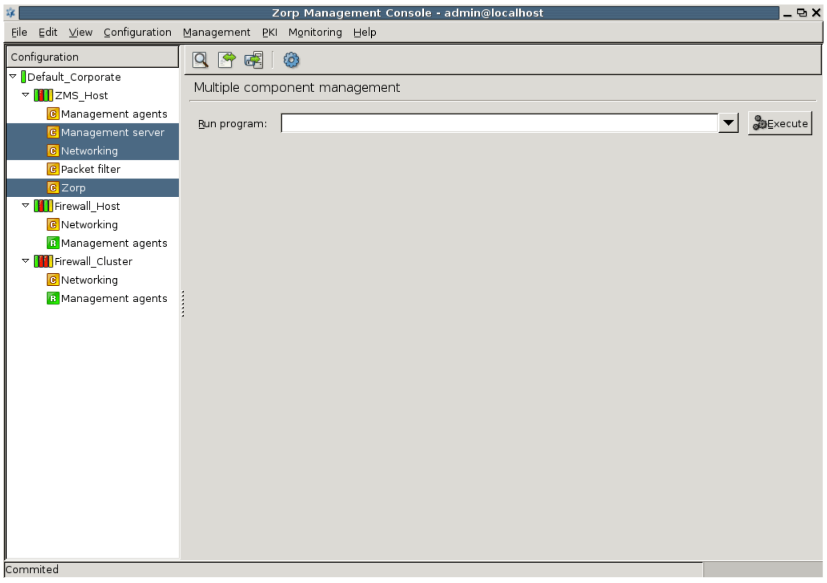


*Figure 3.26. Selecting multiple components in the **Configuration** tree*

3.3.8. Links and variables

There are two options to refer to components involved in network configurations.

- Create links.
- Use variables.

If links are used, IP addresses can be entered manually or select the link target from a drop-down menu. Using links has the advantage that future changes in the network setup do not influence the operability of the connection.

Delete the existing links with the **Unlink** and **Unlink as value** options.

- **Unlink** removes the link connection, meaning that the link field is left empty.
- **Unlink as value** deletes the link but leaves the target IP address in the field which will then behave as a manually added address.

Components can be referenced with the help of variables. By using variables values appearing in several places can be changed at once. If a variable is modified, all corresponding values are changed. Variables are denoted with $ characters preceding and following their names.

Example 3.1. Referring to components with variables
The following is an example for a variable:

```
$autobind-ip$
```

3.3.9. Disabling rules and objects

During the management and maintenance of the firewall host it is often useful to be able to temporarily turn off certain rules, policies, and so on. In Zorp this feature is implemented via the **Disable**/**Enable** options of the local menus. To display the local menu of a rule or object, right-click on the object. For example, a packet filter rule that is only rarely used can be simply disabled when it is not required, to be enabled again when it is required. Disabled rules and objects are generated into the configuration file as comments with the # prefix.

Disabled objects can be edited, modified similarly to any other objects. However, their validity (whether for example, the required parameters are filled, their name is unique, and so on) is checked only when they are enabled again.

The following objects can be disabled in the various ZMC components:

Host:

- *Quarantine cleanup rules*

Packet filter:

- ***Rules***
- ***Groups***
- ***Tables***
- ***Chains***

Disabling a group automatically disables its childrens as well.

Note
Generated rules do not remain disabled after skeleton generation.

Zorp:

- ***Instances***
- ***Rules***
- ***Policies*** (including *matchers*)

Networking:

- ***Interfaces***

- ***Routes***

Date and time:

- ***NTP time servers***

Content vectoring:

- ***Routers** and **rule groups***

ZAS:

- **Routers**

Heartbeat:

- **Resources**

IPSec VPN:

- **Connections**

Mail transport:

- ***Listen interfaces***
- **Transport maps**
- **Virtual maps**
- **Sender address restrictions**
- **Recipient address restrictions**

3.3.10. Filtering list entries

ZMC displays information in several places as tables of entries having various parameters or meta-information. Such filter windows are used to display *firewall rules*, *Zorp logs*, *Active connections*, and so on. The common properties and handling of these tables is summarized in this section.

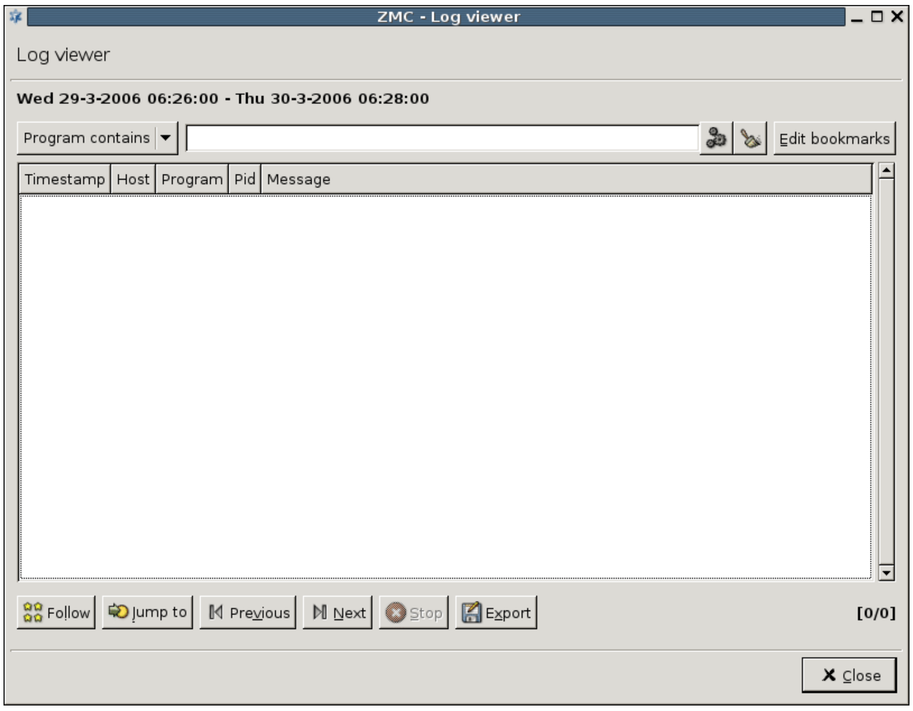


Figure 3.27. A filter window

Filter windows consist of three main parts:

- Filter bar
- Table displaying the entries
- Command bar to perform various actions on the selected entries

The actions available in the command bar are described at the documentation of the actual component using the filter window (for example, *Log viewer*).

Each entry of the table consists of a single row, with the various parameters displayed in labeled columns.

- To sort the entries, click on the column headers.
- To modify the order of the columns, drag the column header to its desired place.
- To hide a single column, right-click on the column header and click **Hide This Column**.
- To configure the columns to be displayed, right-click on the column header and click **Set columns...**, select a column and move it with the arrow. Click **OK**.

To filter the entries, use the filter bar located above the table. The **Filter type** specifies the column to search for the expression (string or regular expression) typed in the field. To search, click **Filter now**. To restore the full list, click **Clear**. To create custom filter expressions, select the **Advanced** option as **Filter type**. The filter expressions can also be combined using the logical `AND` (**if all criteria are met**) and `OR` (**if any criteria are met**) operations. Custom filters can be run once (click **Ok**), or they can be bookmarked for repeated use (click **Save**).

Tip
To display the **Advanced filter editor** dialog with the latest advanced filter, click **...**.

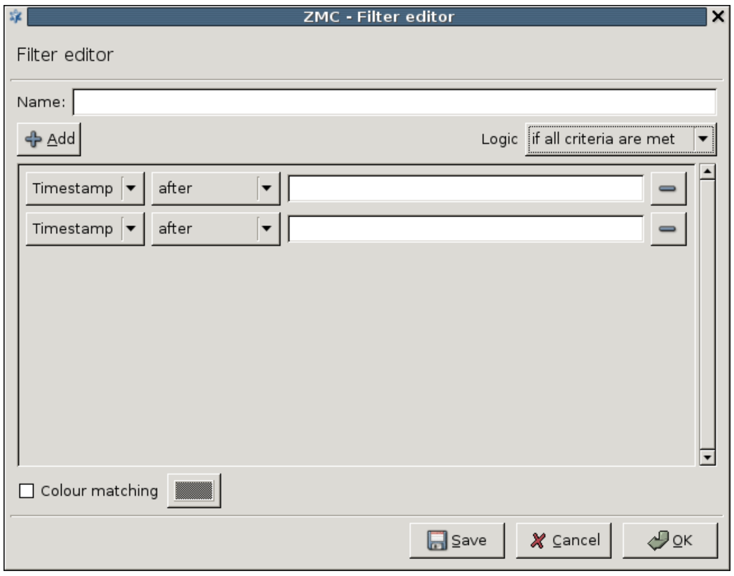


Figure 3.28. Advanced filtering

Saved filters are also displayed in the **Filter type** combobox. To manage saved filters (delete, rename and edit), click **Edit bookmarks**.

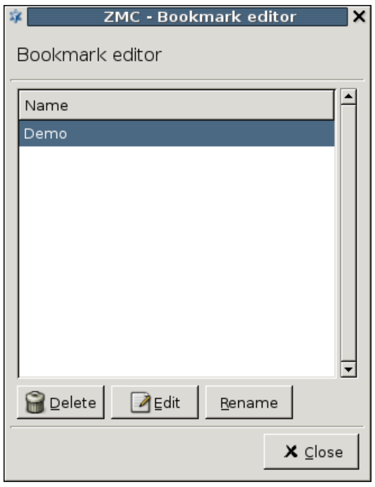


Figure 3.29. Editing bookmarks

In some cases (for example, in the *Log viewer*), advanced filters can be configured to highlight the entries matching the filter expression. To configure highlighting, select **Color matching** and set the color of the highlighting.

Tip
By assigning different colors to different bookmarked filters, the important elements of the table can be highlighted in several colors.

3.4. Viewing Zorp logs

Logs provide an interface for inspecting the log messages collected on a host. To view the logs of a host, select the **Host** and click **View log**.

Tip
Zorp can also create reports about the transferred traffic. For details, see *Section 6.9, Traffic reports (p. 188)*.

The log viewer interface consists of a filter bar to select the messages to be displayed, a list of the actual log entries (including meta-information such as timestamp, and so on), and a *command bar*.

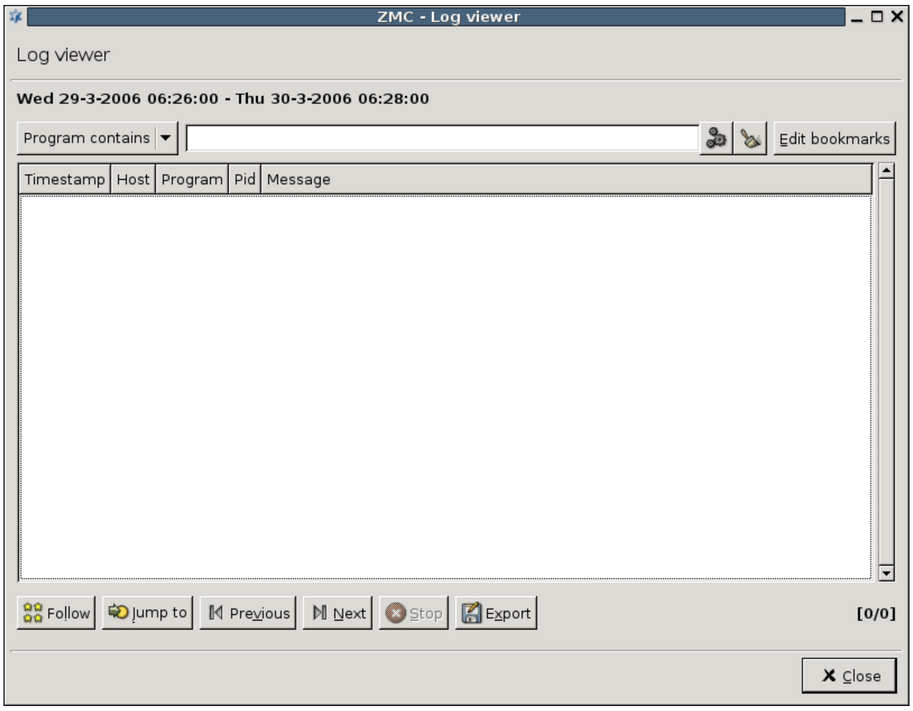


Figure 3.30. Viewing logs

The following information is displayed about the messages:

- **Timestamp**: It shows the exact date when the message was received.
- **Host**: It is the host sending the message.
- **Program**: It is the application sending the message (for example, cron, zms-engine, and so on).
- **Pid**: It is the Process ID of the application sending the message.

- **Message**: It is the message itself.

The **Log viewer** window is a *Filter window*. Therefore, various simple and advanced filtering expressions can be used to display only the required information. For details on the use and capabilities of *Filter windows*, see *Section 3.3.10, Filtering list entries (p. 48)*.

3.4.1. The command bar of the log viewer

The command bar offers various operations to display and export the logs of the host. The following operations are available:

- **Follow**: Start the follow mode and monitor the log messages real-time. The list is updated every second.
- **Jump to**: Select a time interval and display the log messages received within this interval. To specify an interval, enter its **Start date** and **Start time**, and either both **End date** and **End time**, or the **Interval length**. Enter the name of the log file in the **Log file** field.

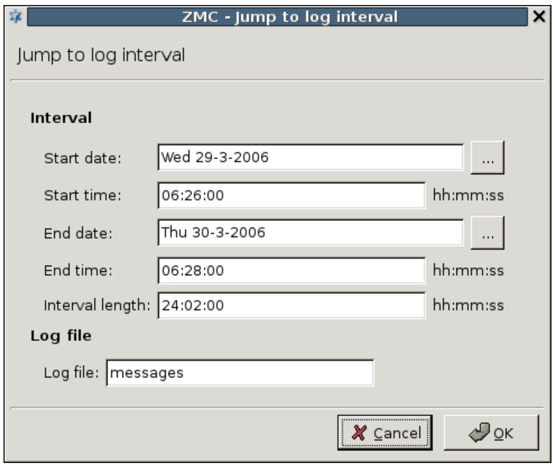


Figure 3.31. Selecting the log interval to be displayed

- **Previous**: It displays the log messages of the previous period (using the same interval length).
- **Next**: It displays the log messages of the next period (using the same interval length).
- **Stop**: It stops the **Follow** mode.
- **Export**: It exports the currently displayed log messages into a file on the local machine that is running ZMC, using either plain text or `CSV` file format.
- **Type of messages**: The rightmost combobox selects the type of messages that are displayed (for example, `zms` , `packet filter`, and so on).

Chapter 4. Registering new hosts

Zorp and ZMS can be used in several network scenarios. In the simplest case there is only a single firewall host having both Zorp and ZMS services installed. In this case, the communication between ZMS and the Zorp management agents takes place locally, using Unix domain sockets and it does not require network communication setup. However, when the two functions, that is, firewalling and management, are separated and installed on two different machines, the initial communication channel between the two requires manual setup. After successful setup all further communication is initiated automatically without manual interaction. This channel setup is a one-time action, therefore it must be configured separately for each new Zorp firewall under the authority of a ZMS host. This process is called bootstrapping and can be performed similarly to running a wizard. By the end of the bootstrapping process, the new host is added to the host configuration database of the ZMS host machine.

The connection between ZMS and Zorp can be established in the following ways:

- using bootstrap
- manually through the **Recovery Connection** function
- completely manually

Bootstrapping a Zorp host is one of the most simple methods. Bootstrapping is similar to running a wizard, that is, answering questions and allowing the wizard to carry out the necessary configurations. Alternatively, the connection can be established manually. This method may especially be needed in troubleshooting scenarios with the help of the **Recovery Connection** button. Hosts can be added on a completely manual way, by selecting a site and then clicking **Add** in the main workspace. For more details, see the *Zorp Professional 7 Reference Guide*.

4.1. Procedure – Bootstrap a new host

Purpose:

To bootsgrap a new host, complete the following steps.

Steps:

Step 1. Select the desired site in the configuration tree.

Step 2. Click **Bootstrap** at the bottom of the screen. This will start the wizard.

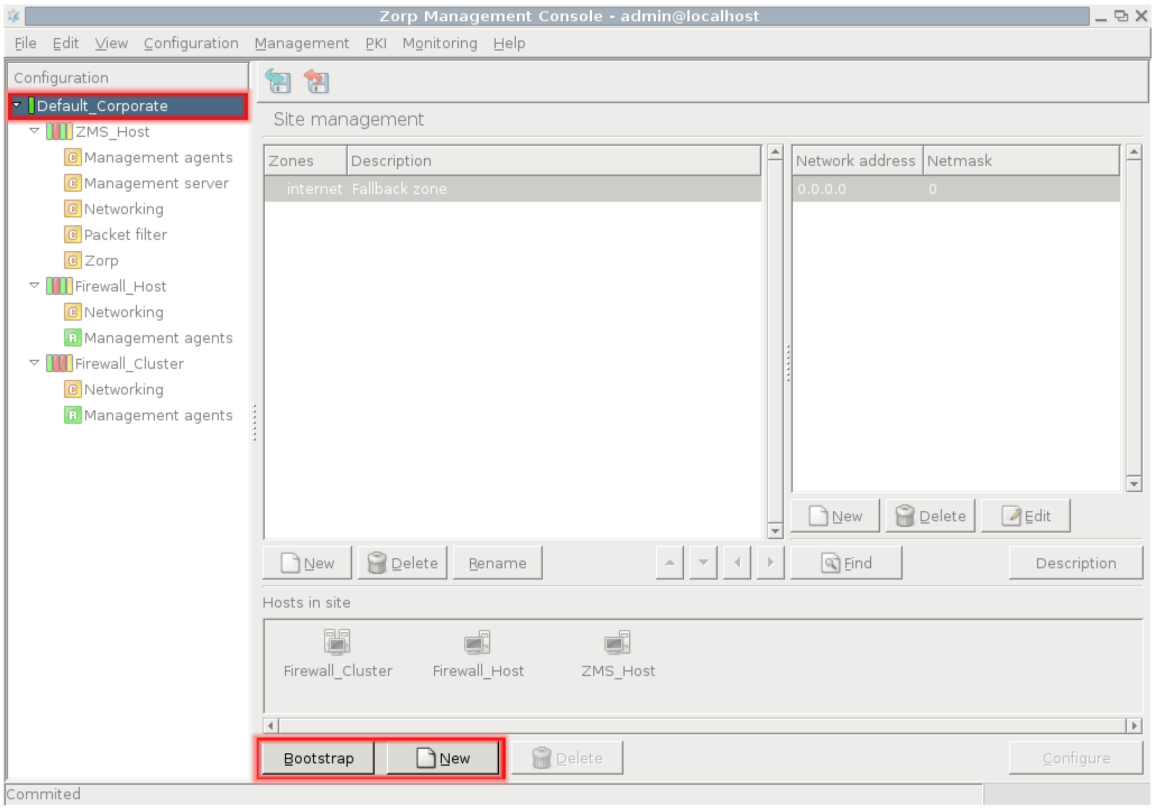
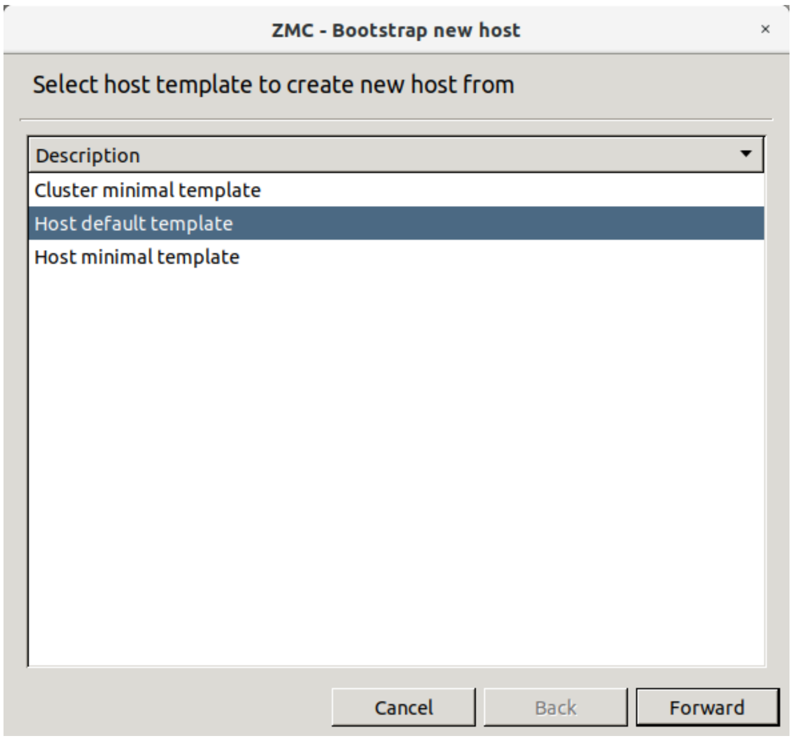


Figure 4.1. Bootstrapping a new host

Note
Alternatively, to register a host manually, click **New** next to **Bootstrap**.

Step 3. Select a host template.



Select host template to create new host from

Description
Cluster minimal template
Host default template
Host minimal template

Figure 4.2. Selecting a host template

The default templates are the following. New templates can be created later.

- **Cluster minimal template**
  This tenplate can be used to configure clustered solutions.

- **Host default template**
  This template can be used to add the **NTP**, **Zorp** and **Packet filter** components automatically to the **Configuration** tree under the name of the newly added host.

  For details, see:

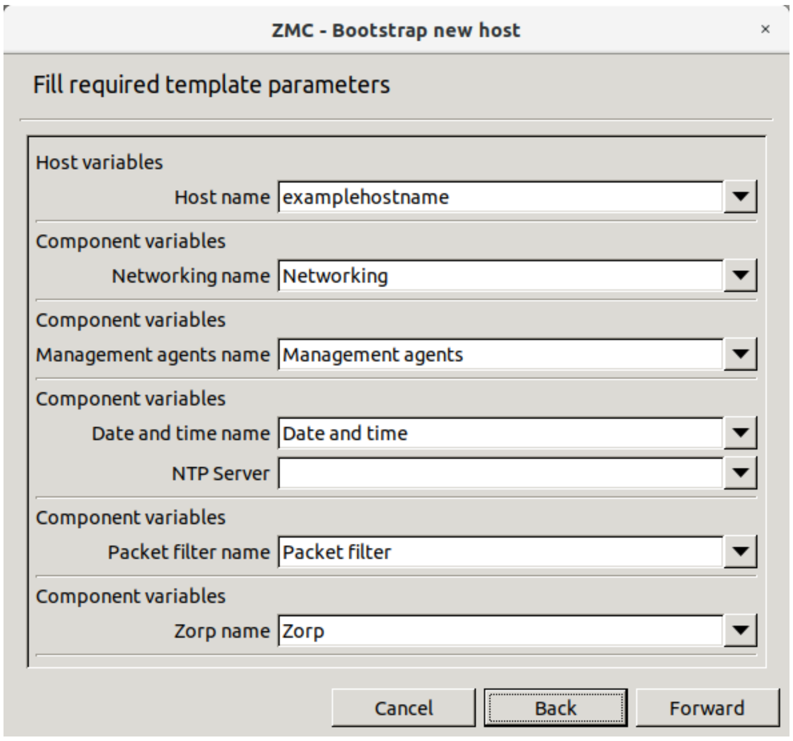
  - **NTP**: *Chapter 9, Native services (p. 218)*
  - **Zorp**: *Chapter 6, Managing network traffic with Zorp (p. 87)*
  - **Packet filter**: *Appendix A, Packet Filtering (p. 462)*

- **Host minimal template**
  This template can be used to add only the two default components: **Management agents** and **Networking**.

It is recommended to select the **Host default template** because it already has some components preconfigured.

Tip
When working with several Zorp hosts it can be useful to create predefined templates, to save repetitive work. For details on creating templates manually, see *Chapter 6, Managing network traffic with Zorp (p. 87).*

Step 4. Enter the details of the new host.



Fill required template parameters

Host variables
Host name: examplehostname

Component variables
Networking name: Networking

Component variables
Management agents name: Management agents

Component variables
Date and time name: Date and time
NTP Server:

Component variables
Packet filter name: Packet filter

Component variables
Zorp name: Zorp

Figure 4.3. Parameters of the new host

Host name: It is the name of the Zorp firewall.

Networking name: It is the name of the **Networking** component.

Management agents name: It is the name of the **Management agents** component.

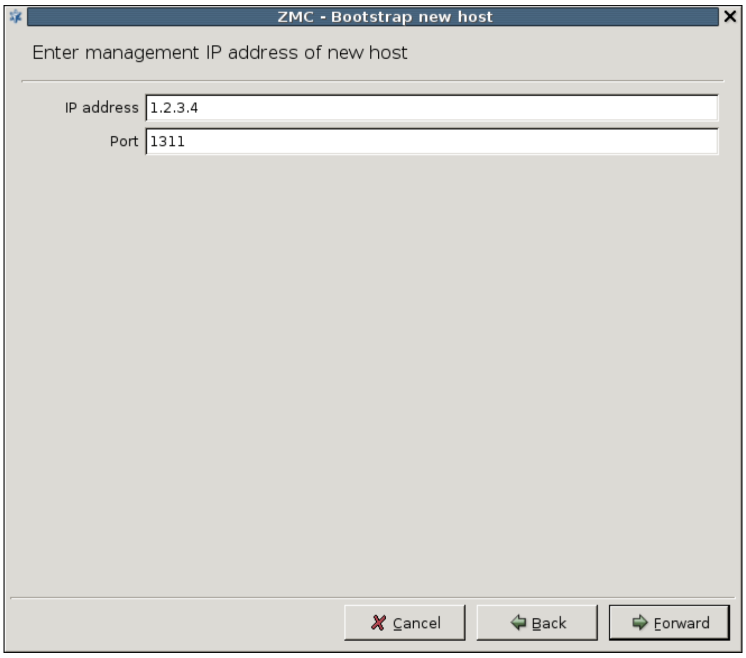
Date and time name: It is the name of the **Date and time** component.

NTP Server: Specify a time server that Zorp synchronizes its system time with. Usually, but not necessarily it is an external time source. For the up-to-date list of publicly available time servers, see *http://support.ntp.org/bin/view/Servers/WebHome*. For more information on NTP, see *Chapter 9, Native services (p. 218).*

Packet filter name: It is the name of the **Packet filter** component.

Zorp name: It is the name of the **Zorp** component.

Step 5. Enter the IP address and configuration port number of the Transfer agent.



Enter management IP address of new host

IP address 1.2.3.4

Port 1311

Figure 4.4. Entering the management IP address of the host

Before starting, discover which network interface and IP address is reachable from the network location. Firewalls almost always have more than one of these. Ensure that the IP address typed in is reachable from the current location and that the packets will find their ways back from the firewall. In other words, make sure that all routing information is correctly configured.

YConfigure other interfaces of Zorp to be reachable for configuration purposes later.

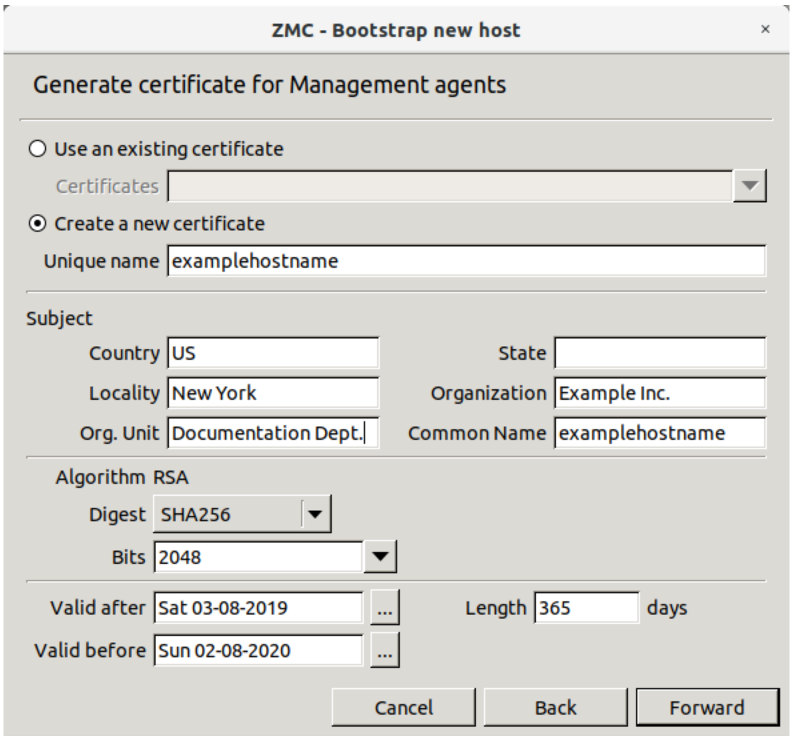
Step a. Enter the **IP address**.
Refer to the Firewall's installation documentation for the IP address information.

Step b. Leave the **Port** field default (`1311`).

Step 6. Create a certificate for SSL communication establishment.
Firewalls are administered from a protected, inside interface and while this method is highly recommended, it is not necessarily required. All the configuration traffic is encrypted, using SSL.

The administrative connection is encrypted using SSL, which requires a certificate, especially the public key it contains. This certificate and the private key used for encryption/decryption are sent to the Management agent on the firewall node that uses it to encrypt the session key it generates. For more information on SSL communication establishment, see *Chapter 11, Key and certificate management in Zorp (p. 249)*.

Enter the parameters of the certificate and it will be generated automatically.


ZMC - Bootstrap new host

Generate certificate for Management agents

Use an existing certificate [✓ Create a new certificate]

Certificates			
Unique name	examplehostname		
Subject			
Country	US	State	
Locality	New York	Organization	Example Inc.
Org. Unit	Documentation Dept.	Common Name	examplehostname
Algorithm	RSA		
Digest	SHA256		
Bits	2048		
Valid after	Sat 03-08-2019	Length	365 days
Valid before	Sun 02-08-2020		

Figure 4.5. Creating a certificate for SSL communication establishment

Step a. To **Create a new certificate**, enter a name for the certificate in the **Unique name** field. Alternatively, to **Use an existing certificate**, browse for a certificate from **Certificates**. The following steps describe the details of creating a new certificate.

Tip
There are no particular requirements for the **Unique name** and **Common Name** fields other than trivial string length and restricted character issues. However, it is recommended to enter a name that will later — when there are more certificates in use in the system — uniquely and easily identify this certificate as the one used for establishing agent communication.

Step b. In the **Country** field, enter the two-character country code of the country where Zorp is located. For example, to refer to the United States, enter `US`.

Step c. *Optional step*: In the **State** field, enter the state where Zorp is located, if applicable. For example, `California`.

Step d. In the **Locality** field, enter the name of the city where Zorp is located. For example, `New York`.

Step e. In the **Organization** field, enter the name of the company that owns Zorp. For example, `Example Inc.`.

Step f. In the **Org. Unit** field, enter the department of the company that administers Zorp. For example, `IT Security`.

Step g. Enter a **Common Name** that describes you or your subdivision.
Alternatively, a default value can be used: the name of the Zorp firewall node.

Step h. Configure the RSA algorithm. Select whether to use `SHA-256` or `SHA-512` **Digest**. Select the asymmetric key length from the **Bits** list.

> **Note**
> The U.S. National Institute for Standards and Technology (NIST) recommends `2048`-bit keys for RSA.

Step i. Configure the validity range of the certificate.

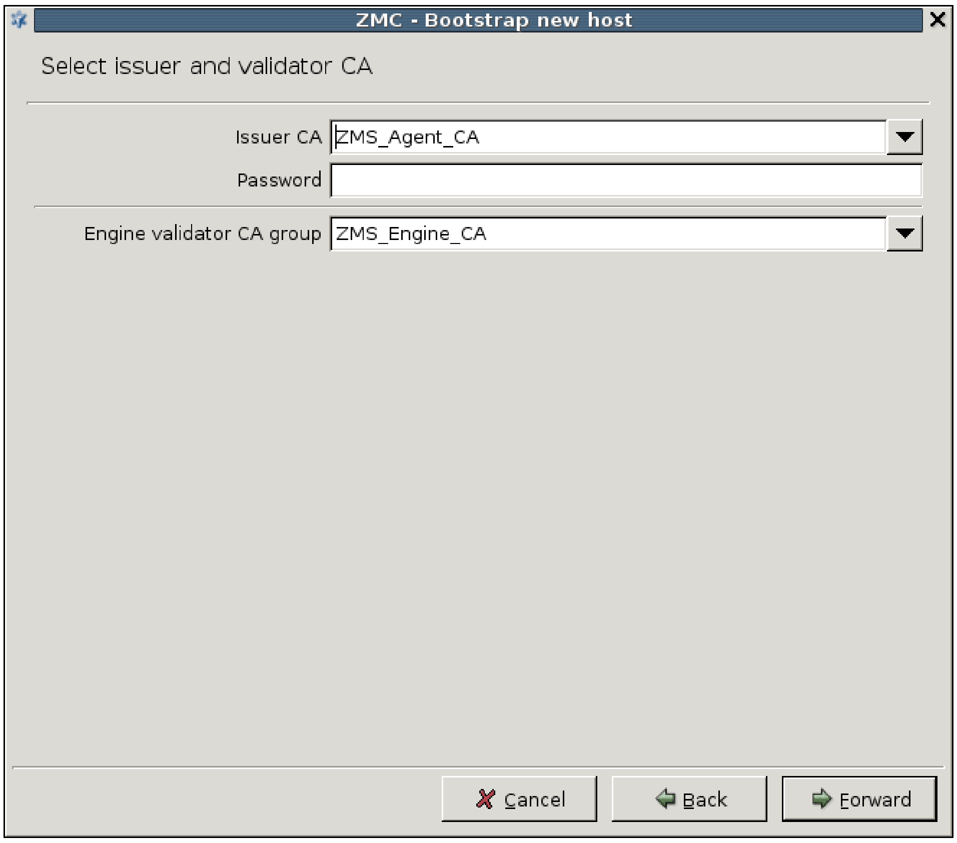
To select the start date from the **Valid after** field, click ....

To configure the validity range, either select the end date from the **Valid before** field by clicking ... or enter the validity **Length** in days and press **Enter**.

Step 7. Enter the ZMS Agent CA password.
Manually entered passwords protect private keys against possible unauthorized accesses. Even if an attacker has read access to the hosts, the private keys cannot be stolen (read). These passwords are used to encrypt the private keys and therefore they are never stored in unencrypted format at all. Certificates are issued by Certificate Authorities (CA) and it is actually the CA's private key that requires this protection. The certificate used by the Management agents are issued by the ZMS_Agent_CA. Enter the password for this CA that was defined for this purpose when the ZMS service was installed. See also *Chapter 11, Key and certificate management in Zorp (p. 249)*.

> **Note**
> To generate a strong password, it is recommended to use a password generator.

> **Tip**
> Take detailed logs of the installation process, including the bootstraps where all these passwords are recorded.


ZMC - Bootstrap new host

Select issuer and validator CA

Issuer CA: ZMS_Agent_CA

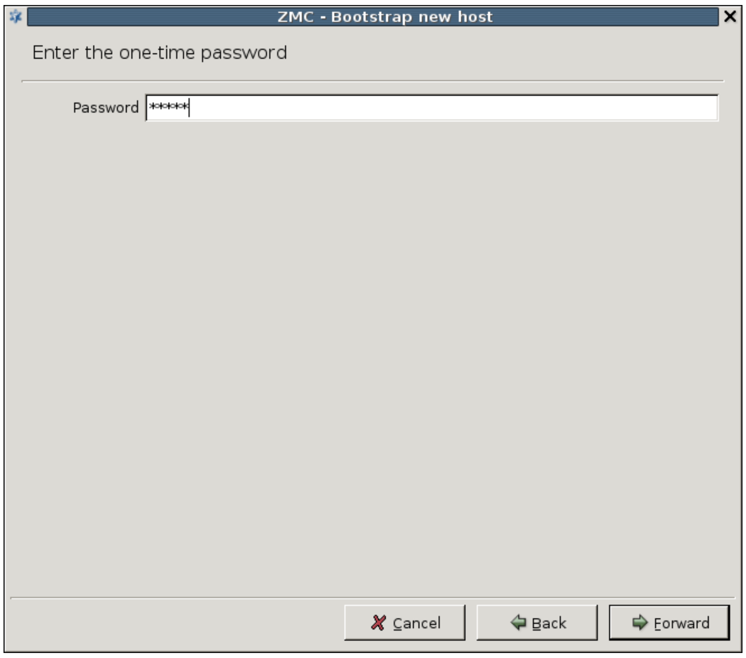
Password:

Engine validator CA group: ZMS_Engine_CA

Figure 4.6. Entering ZMS Agent CA password

Step 8. Enter the One-Time Password.

The **one-time password** is the one that has been entered during the installation of ZMS-transfer-agent on the Zorp host. It is a one-time operation: to establish an SSL channel between the Management agents of Zorp and the ZMS host, certificates are required. There are no certificates to use, therefore a certificate has to be provided for the ZMS-transfer-agent on Zorp, which can be used for communication channel buildup purposes. This password is used to establish a preliminary encrypted communication channel between Zorp and the ZMS host, where the certificate can be sent. All communication among the parties is performed using SSL.


ZMC - Bootstrap new host

Enter the one-time password

Password *****

Figure 4.7. Entering the One-Time Password

Step 9. Click the final **OK** button if all password phases have been successful to build up the connection. The displayed logs provide information about the steps the wizard takes in the background. To save the output for later analysis, if needed, (either by the administrator or a support personnel), click **Save**.

> **Note**
> If anything goes wrong, the wizard returns to the window where the mistake was made, so that it can be corrected.

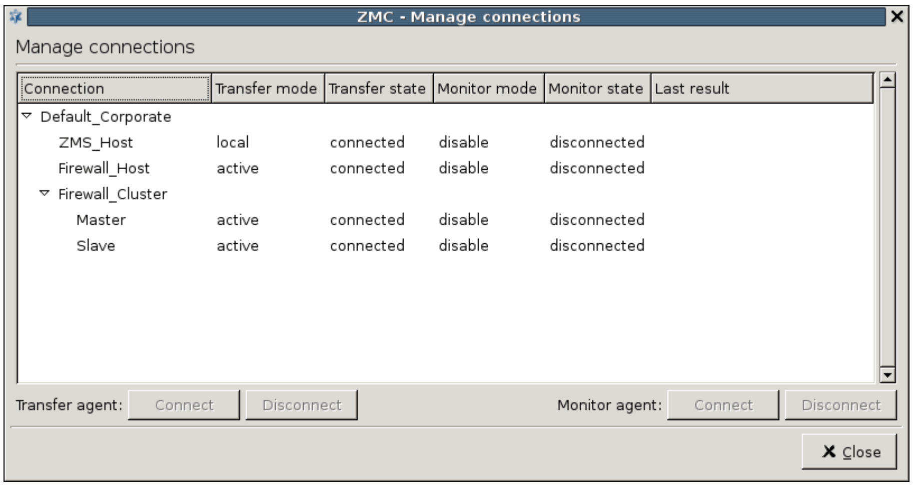
After the bootstrap process has finished successfully, the new host is ready to be configured.

4.2. Reconnecting to a host

When starting ZMC and selecting a host in the Configuration tree, the connection with the host is automatically established. If, for some reason it breaks, the host has to be reconnected manually.

4.2.1. Procedure – Reconnecting ZMS to a host

Step 1. Navigate to Management > Connections....


ZMC - Manage connections

Manage connections

Connection	Transfer mode	Transfer state	Monitor mode	Monitor state	Last result
Default_Corporate					
ZMS_Host	local	connected	disable	disconnected	
Firewall_Host	active	connected	disable	disconnected	
Firewall_Cluster					
Master	active	connected	disable	disconnected	
Slave	active	connected	disable	disconnected	

Figure 4.8. Managing connections manually

This window accurately shows that it is not the Zorp host that directly communicates with ZMS, but the Management agent installed on it.

Agents are responsible for reporting firewall configuration and related information to the ZMS and are also responsible for accepting and executing configuration commands. Communication between the Transfer Agent and ZMS uses TCP port `1311`. The Transfer Agent must be installed on all firewall nodes to be managed with ZMS. By default, ZMS establishes the communication channel with the agents, but the agents can also be configured to start the communication if required.

Step 2. Connect and/or disconnect the appropriate agents with the corresponding buttons.

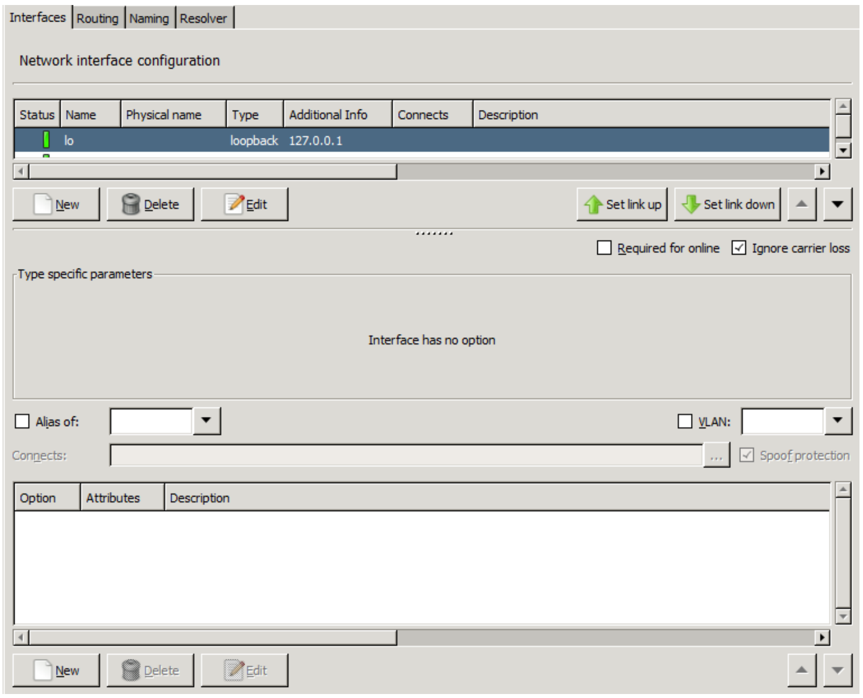
Chapter 5. Networking, routing, and name resolution

ZMS is a complex central management facility for Zorp firewalls. Besides firewall-centric configuration settings, such as firewall policies, packet filter rules, it allows for the configuration of several basic parts of the operating system. In fact, one of the design goals of ZMS was to eliminate the need for command-line configuration of the operating system and Zorp as much as possible. Therefore tools are provided to perform basic, operating system-level configuration tasks.

The **Networking** component that is present by default for each host in the **Configuration** tree serves this purpose by providing access to all the relevant network-related configuration areas of the host's operating system. The possible settings in the **Networking** component are mostly related to ordinary network configuration issues and there are hardly any variables directly related to firewalling functions.

The main window of the **Networking** component is divided into the following four tabs.

- *Interfaces tab* for configuring general interface-related settings
- *Routing tab* for managing network routes
- *Naming tab* for configuring name resolution
- *Resolving tab* for configuring client-side DNS resolution



Status	Name	Physical name	Type	Additional Info	Connects	Description
	lo		loopback	127.0.0.1		

Option	Attributes	Description

Figure 5.1. Tabs in the **Networking** component

Warning
It is recommended for the user not to create any files with the '00-zms' prefix to the Networking component, that is to the /etc/systemd/network/ because the ZMS GUI might handle these files and will probably either modify or delete them.

5.1. Configuring networking interfaces

The configuration of network interfaces can be performed on the **Interfaces** tab of the **Networking** ZMC component. These tasks fall into the following categories.

- *General interface configuration*
- *Configuring virtual networks and alias interfaces*
- *Enabling spoof protection*
- *Configuring interface options and activation scripts*

General and special interface configuration features are available for every host managed from ZMS, while spoof protection is intended for Zorp gateways or other hosts that have an active packet filter installed.

5.1.1. General interface configuration

The **Interfaces** tab of the **Networking** ZMC component lists the network interfaces available on the host, along with their type, IP addresses, and connected zones.

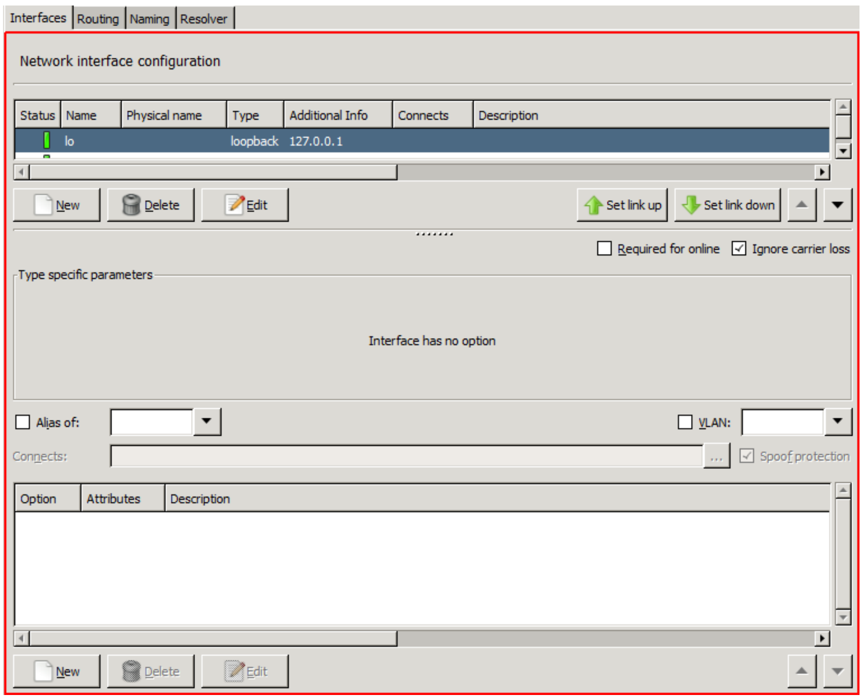


Figure 5.2. Network interface configuration

Tip
If no one or more physical interfaces of the host are listed here, it is most likely because they were not configured before bootstrapping took place. The bootstrapping process not only establishes the connection between ZMS and the host (Management agents on it), but it also queries host configuration, and inserts this information into the ZMS database. When selecting a host entry in ZMC, the information is read from the ZMS database, and not from the host directly. Therefore, ZMC does not detect parameters that were unavailable for ZMS during bootstrapping.

To correct this situation, define the missing interface(s): click **New** and configure them as required.

5.1.1.1. Procedure – Configuring a new interface

Purpose:

To define a new interface, complete the following steps.

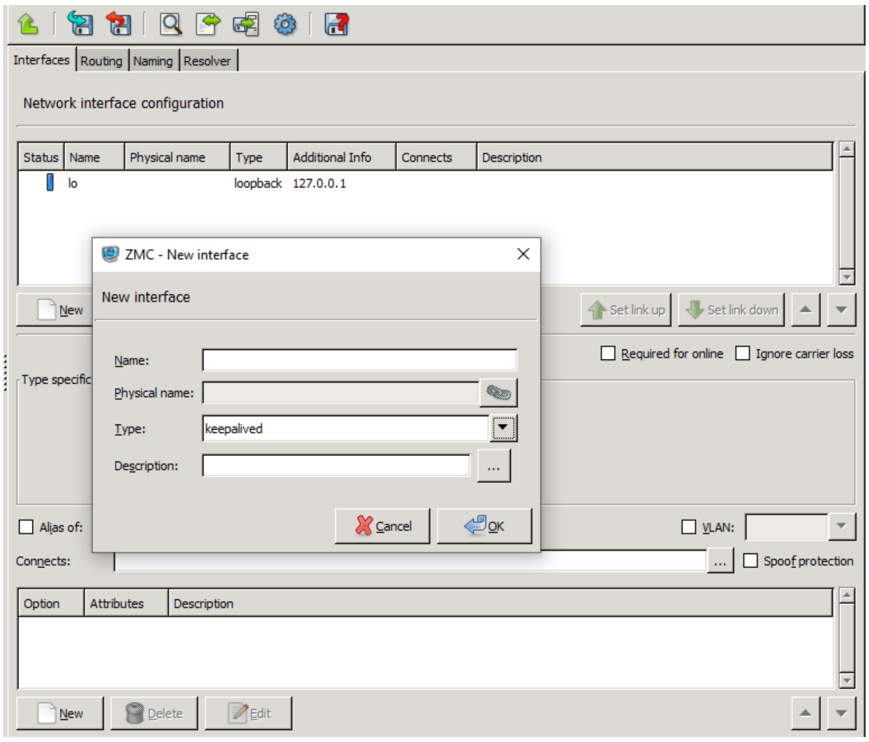
Steps:

Step 1. Navigate to **Networking** > **Interfaces** tab.

Step 2. Click **New**.

Step 3. In the **Name** field, enter a name for the interface (for example, `eth0`).

Step 4. Select the **Type** of the interface.



Network interface configuration

Status	Name	Physical name	Type	Additional Info	Connects	Description
	lo		loopback	127.0.0.1		

ZMC - New interface

New interface

Name:
Physical name:
Type: keepalived
Description:

Required for online
Ignore carrier loss
Type specific
Alias of:
VLAN:
Connects:
Spoof protection

Option	Attributes	Description

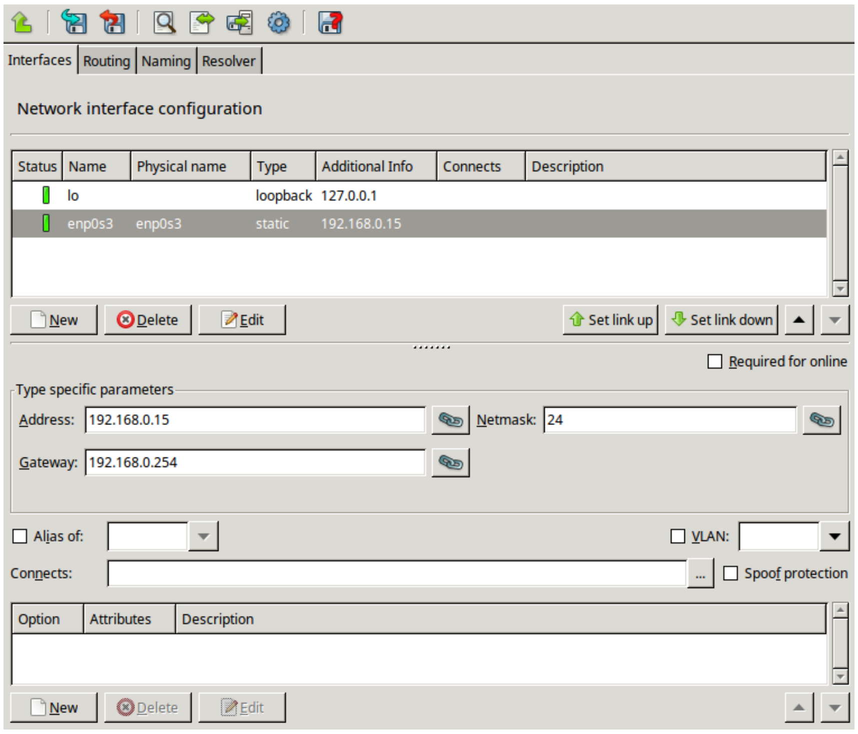
Figure 5.3. Defining a new interface

Step 5. *Optional step*: Enter a description, if required. To enter a longer description, click ....

Step 6. Click **OK**.

Step 7. Configure the parameters of the interface below the table: enter the **IP Address**, **Netmask**, **Gateway Address**, and other data as required. The list of type-specific parameters depend on the type of interface being configuring.
For static interfaces, that is, regular Ethernet interfaces enter the `Netmask` parameter using the CIDR notation.

Warning
As with all firewalls, only one gateway address can be specified in the network configuration and only for a single interface. The gateway box for all other interfaces must be empty.



Network interface configuration

Status	Name	Physical name	Type	Additional Info	Connects	Description
🟩	lo		loopback	127.0.0.1		
🟩	enp0s3	enp0s3	static	192.168.0.15		

Required for online: ☐

Type specific parameters

Address: 192.168.0.15 Netmask: 24

Gateway: 192.168.0.254

☐ Alias of: ☐ VLAN:

Connects: ☐ Spoof protection

Option	Attributes	Description

Figure 5.4. Configuring a new interface

Note
If the configuration information entered in ZMC is not the same as the current settings on the host, the settings of the host are overwritten during the next **Upload** action.

Step 8. Check in (or leave checked in) the *Ignore carrier loss* parameter if it is required to enable the interface to retain its configuration even in case there is temporarily no carrier for it.
This value is checked in by default. It is strongly recommended to keep this value checked in.

If the interface settings are changed, in order to activate the changes, restart the modified network interface. Additionally, it may be required to temporarily stop an interface for security or maintenance reasons with the **Actions** button under the network interface listing.

Warning
Restarting the interface might terminate all ongoing connections of the interface.

Note
The interfaces are controlled individually.

5.1.1.2. Dynamic interfaces

Dynamic interfaces are interfaces that are either created dynamically, or obtain IP configuration information dynamically from a designated server (for example, *dhcp*, *bootp*, *ppp*). As their IP configuration is not known when Zorp boots up (and can be different at each boot sequence), the services using these interfaces cannot include the IP address of the interface in the firewall rules related to the service. To overcome this problem, Zorp can bind to interfaces instead of IP addresses. Dynamic interfaces are referenced by their name in the firewall rules. The operating system automatically notifies the running Zorp instances when the IP configuration information of the interface is received from the server. IP address changes are also automatically handled within Zorp. For more information on configuring firewall rules, see *Section 6.5, Configuring firewall rules (p. 135)*.

Example 5.1. Referencing static and dynamic interfaces in firewall rules
Dynamic interfaces can be used in firewall rules the same way as static interfaces. The following rule references a static interface:

```
Rule(proto=6,
    dst_iface='eth0',
    service='test'
    )
```

The following rule references a dynamic interface called *dyn*:

```
Rule(proto=6,
    dst_iface='dyn',
    service='test'
    )
```

5.1.2. Configuring virtual networks and alias interfaces

In some cases it can be useful to fine-tune the network for special purposes. For example for Virtual Local Area Network (VLAN) technology: many organizations use it for security and network traffic separation purposes. VLANs are logically separated components of physical networks. Logical separation means that although they are on the same physical network (otherwise known as broadcast domain) hosts on separate VLANs cannot communicate with each other unless a router is set up that provides the interconnection. Routing functions for VLAN, and VLAN creation in general, are typically performed by Layer 3 Ethernet switches. Provided that VLAN-capable network cards are installed in the machine, Zorp fully supports VLANs and ZMS provides a control for configuring it.

VLAN interfaces are named in the following manner:

ethx.n where

x	is the number of the physical interface
n	is the ID of the VLAN. The ID of the VLAN is usually a number (for example, 0 for the first VLAN of the interface, 1 for the second, and so on).

Note
If an interface is defined as a VLAN interface, it cannot operate as a real, physical interface at the same time.

For example, the eth1.12 VLAN interface is the 12th VLAN interface of the *eth1* physical network interface. If a VLAN is defined for *eth1*, it cannot be used *eth1* as a physical interface.

5.1.2.1. Procedure – Creating a VLAN interface

Purpose:

To create a VLAN interface, complete the following steps.

Steps:

Step 1. Every VLAN interface must be connected to a physical interface. If not already configured, configure the physical interface that will be used as the VLAN interface. See *Section 5.1.1, General interface configuration (p. 64)* for details.

> **Warning**
> If an interface is defined as a VLAN interface, it cannot operate as a real, physical interface at the same time.

Step 2. To create a new interface, click **New**.

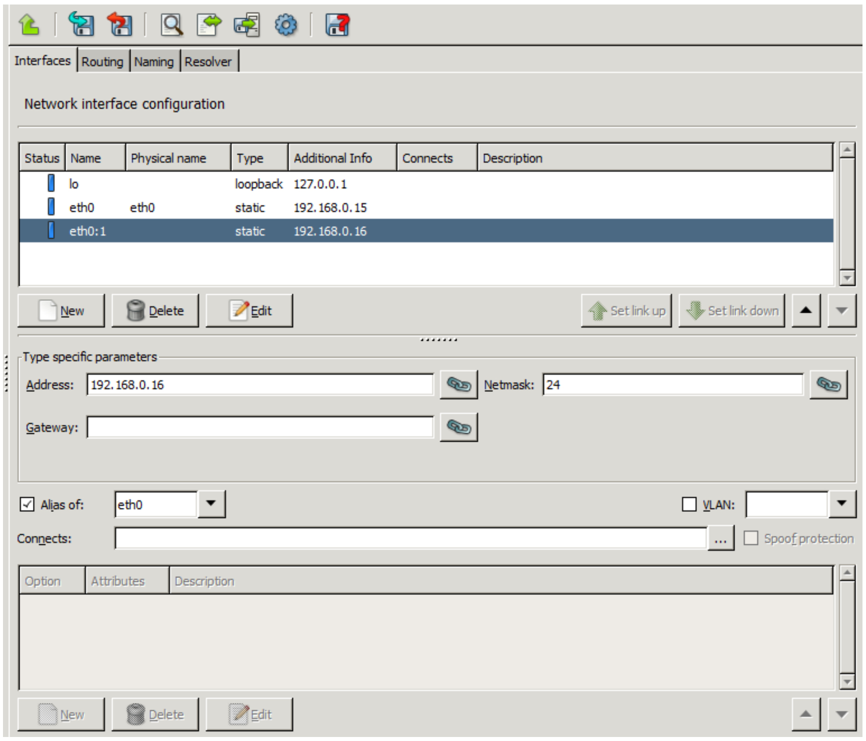
Step 3. Set the **Type** of the interface. The type of the VLAN interface and that of the physical interface can be different.

Step 4. Enter a name for the VLAN interface and click **OK**. The name must include the name of the physical interface, the period (`.`) character, and a number that identifies the VLAN interface (because a physical interface can have several VLAN interfaces), for example, `eth1.0`.
ZMC creates the new interface and automatically selects the VLAN option and the sets the parent interface of the VLAN.

Step 5. Configure other options of the interface (for example, connected zones) as needed.

Step 6. To activate the changes, click **Commit** and **Upload**. Then select the physical interface of the VLAN, click **Control service**, and **Restart** the interface.

> **Warning**
> Restarting the interface might terminate all ongoing connections of the interface.



Network interface configuration

Status	Name	Physical name	Type	Additional Info	Connects	Description
▮	lo		loopback	127.0.0.1		
▮	eth0	eth0	static	192.168.0.15		
▮	eth0:1		static	192.168.0.16		

Type specific parameters

Address: 192.168.0.16 Netmask: 24

Gateway:

[✓] Alias of: eth0 ☐ VLAN:

Connects: ☐ Spoof protection

Option	Attributes	Description

Figure 5.5. Configuring VLAN and alias interfaces

Using alias interfaces allows to configure multiple IP addresses to a physical device. Alias interfaces are named in the following manner:

`ethx:n` where

ethx	is the name of the corresponding physical or VLAN interface.
n	is the ID of the alias interface. The ID is usually a number (for example, `0` for the first alias of the interface, `1` for the second, and so on), but it can be a more informative name as well.

An alias can be defined for existing physical and VLAN interfaces.

5.1.2.2. Procedure – Creating an alias interface

Purpose:

To create an alias interface, complete the following steps.

Steps:

Step 1. Every alias interface must be connected to a physical or a VLAN interface. If it is not already configured, configure this interface.

Step 2. To create a new interface, click **New**.

Step 3. Set the **Type** of the interface. The type of the alias interface and that of the physical interface can be different.

Step 4. Enter a name for the alias interface and click **OK**. The name must include the name of the physical or VLAN interface, the colon (`:`) character, and the number or the name that identifies the alias interface (because an interface can have several alias interfaces). For example, `eth1:0`.
ZMC creates the new interface and automatically selects the alias option and sets the parent interface of the alias.

Step 5. Configure other options of the interface (for example, connected zones) as needed.

Step 6. To activate the changes, click **Commit** and **Upload**. Then select the parent interface of the alias, click **Control service**, and **Restart** the interface.

> **Warning**
> Restarting the interface might terminate all ongoing connections of the interface.

5.1.3. Procedure – Configuring bond interfaces

Purpose:

To create a bond interface, complete the following steps. The interfaces used to create the bond interface must be already configured. For details on configuring network interfaces, see *Procedure 5.1.1.1, Configuring a new interface (p. 65)*.

Steps:

Step 1. Navigate to the host, and select the **Networking** ZMC component.

Step 2. To create a new interface, select **Interfaces** > **Network interface configuration** > **New**. The **New interface** dialog is displayed.

Step 3. Enter a name for the interface (for example, `bond0`) and set its type to *`static`*.

Step 4. Select **Type-specific options** > **New**.

Step 5. Select *`bond_slaves`* and enter the name of the interfaces to bond into the **Attributes** field. Use a single whitespace to separate the name of the interfaces (for example, `eth0 eth1`).

Step 6. *Optional step*: Select **Type-specific options** > **New** to set other bond-specific options as needed (for example, *`bond_mode`*). If an option is not listed, type its name into the **Option** field and its value to the **Attributes** field.

Step 7. Configure the other parameters of the interface (for example, address, netmask, and so on) as needed.

Step 8. To activate the changes, click **Commit** and **Upload**.

5.1.4. Procedure – Configuring bridge interfaces

Purpose:

To create a bridge interface, complete the following steps. The interfaces used to create the bridge interface must be already configured. For details on configuring network interfaces, see *Procedure 5.1.1.1, Configuring a new interface (p. 65)*.

Steps:

Step 1. Navigate to the host, and select the **Networking** ZMC component.

Step 2. To create a new interface, select **Interfaces** > **Network interface configuration** > **New**. The **New interface** dialog is displayed.

Step 3. Enter a name for the interface (for example, `bridge0`) and set its type to *`static`*.

Step 4. Select **Type-specific options** > **New**.

Step 5. Select the *`bridge_ports`* option, and enter the name of the interfaces to bridge into the **Attributes** field. Use a single whitespace to separate the name of the interfaces (for example, `eth0 eth1`). To bridge every available interfaces, enter *`all`*.

Step 6. *Optional step*: Select **Type-specific options** > **New** to set other bridge-specific options as needed (for example, *`bridge_stp`*). If an option is not listed, just type its name into the **Option** field and its value to the **Attributes** field. For a list of available options, see the *bridge-utils-interfaces manual page*.

Step 7. Configure the other parameters of the interface (for example, address, netmask, and so on) as needed.

Step 8. To activate the changes, click **Commit** and **Upload**.

5.1.5. Enabling spoof protection

Spoof protection means that the packet filter module of a firewall checks to ensure that packets arriving on an interface have source IP addresses that are legal in networks reachable through that given interface and accepts only those packages that match this criterion.

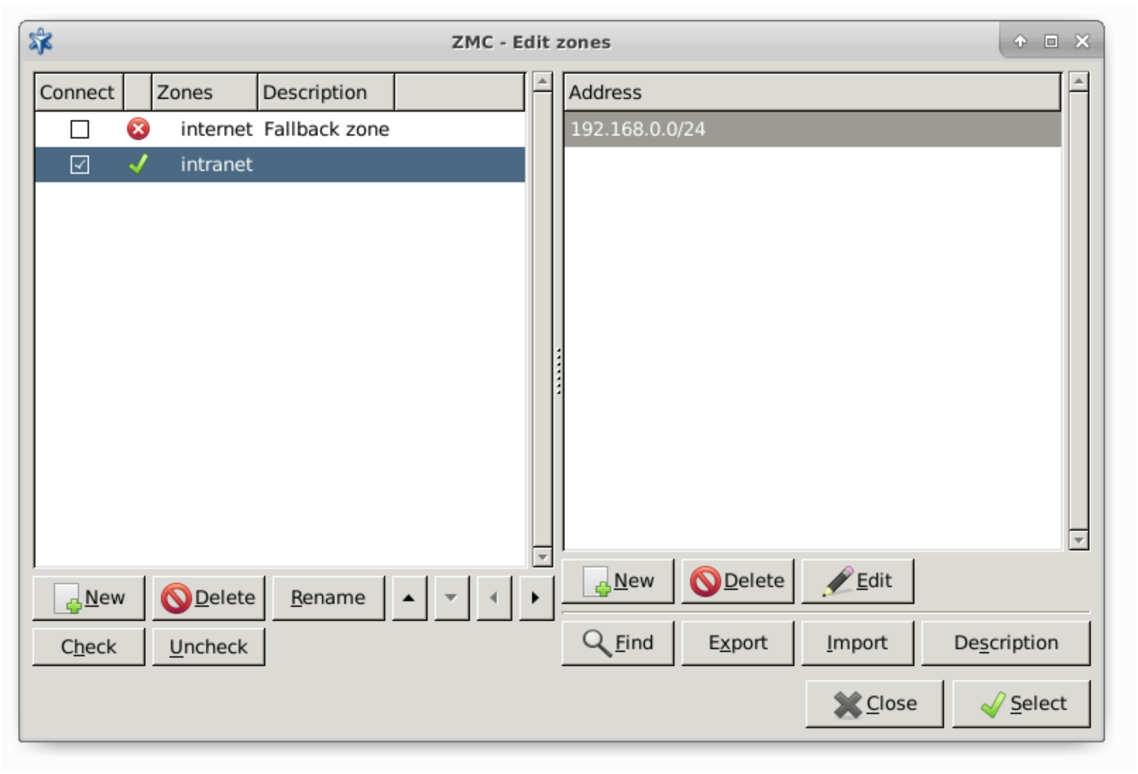
For example, if *`eth0`* connects to the Intranet (`10.0.0./8`) and it is spoof-protected, the firewall does not accept datagrams on this interface with source IP addresses other than the `10.0.0.0/8` range. It does not accept datagrams with source IP address from the `10.0.0.0/8` range on interfaces other than *`eth0`* either.

5.1.5.1. Procedure – Configuring spoof protection

Steps:

Step 1. Select the **Networking** interface and next to the **Connects** field, click ....

Step 2. Select the zones the configured interface is connected to either directly or indirectly through routers.



Connect	Zones	Description
☐	internet	Fallback zone
☑	intranet	

Address
192.168.0.0/24

Figure 5.6. Zone selection for the Connects control

Note
The Zone tree in the dialog window displays organized IP addresses, starting from the most generic (`0.0.0.0/0`) to the most specific. There is an implicit inheritance in the **Connects** specification: if the `10.0.0.0/8` address range is specified to connect to *`eth1`*, all more specific subnets of this address range (`10.0.0.0/9`, `/10`, ...`/32`) connect to *`eth1`* also, unless a more specific binding is explicitly specified.

Step 3. Select **Spoof protection**.

Note
The **Packet filter** ruleset must be regenerated after modifying any interface settings. It is not done automatically.

For further details on zones, see *Section 6.2, Zones (p. 87)*. For more information on Spoof control in relation to packet filter rules, see *Section A.4.3.3, Spoof protection (p. 479)*.

5.1.6. Interface options and activation scripts

The bottom part of the **Interfaces** tab can be used to specify activation scripts and various other parameters of the network interfaces. The following options can be configured:

- *Interface activation scripts*

- *Interface groups*
- *Other interface options*

5.1.6.1. Configuring interface activation scripts

Interface activation scripts can be useful for scenarios where special procedures are required to initialize networking.

For example, changing the Media Access Control (MAC) address of a network card before bringing it up can be done with a pre-up script. Such scripts should also be used for configuring bridge interfaces.

The following types of activation scripts can be set:

1. *post-up* scripts are executed after the interface is activated.
2. *post-down* scripts are executed after the interface is deactivated.

5.1.6.1.1. Procedure – Creating interface activation scripts

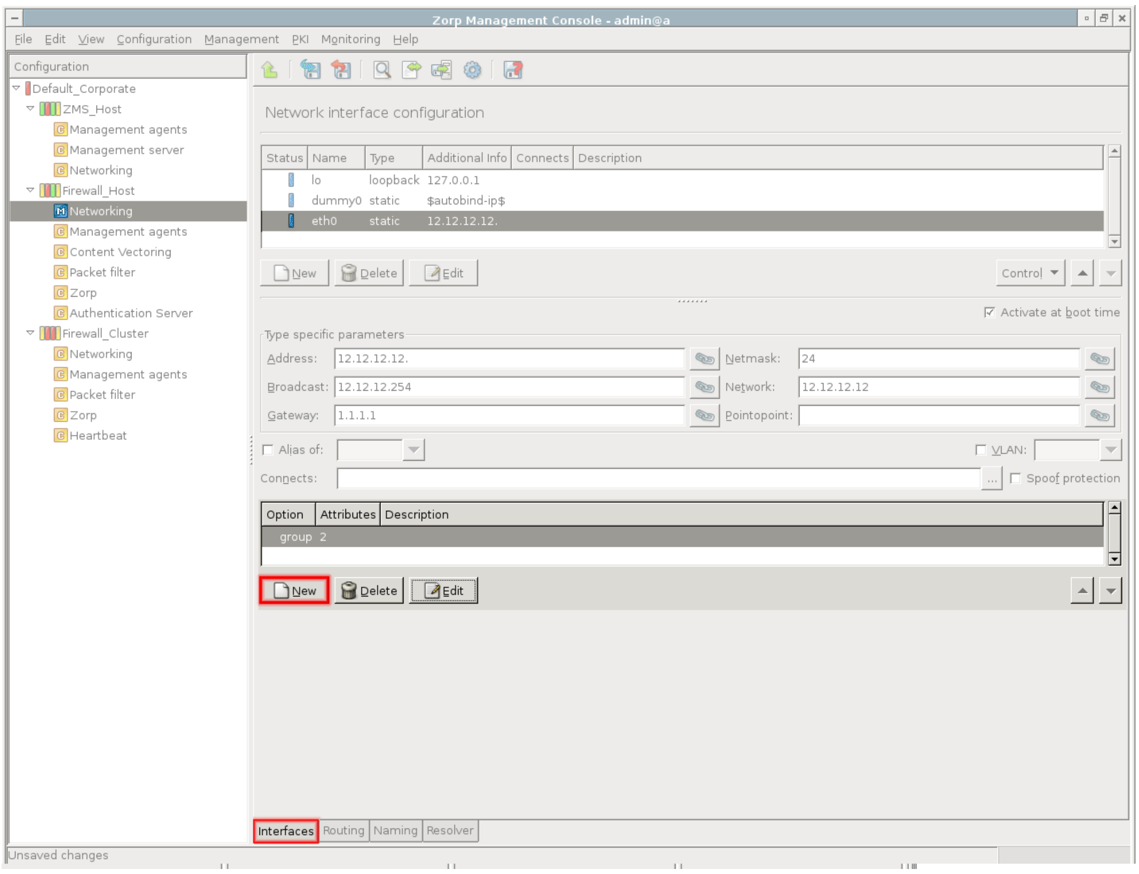
Purpose:

To create an interface activation script for an interface, complete the following steps.

Steps:

Step 1. Navigate to **Networking > Interfaces** and select the interface to configure.

Step 2. To add a new option to the interface, click **New**.



Status	Name	Type	Additional Info	Connects	Description
	lo	loopback	127.0.0.1		
	dummy0	static	$autobind-ip$		
	eth0	static	12.12.12.12.		

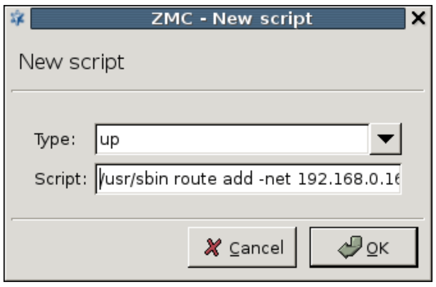
Address:	12.12.12.12.	Netmask:	24
Broadcast:	12.12.12.254	Network:	12.12.12.12
Gateway:	1.1.1.1	Pointopoint:	

Option	Attributes	Description
group	2	

Figure 5.7. Configuring interface options

Step 3. From the **Option** field, select the type of the script (`post-up`, `post-down`).

Step 4. Enter the command to be executed to the **Script** field under **Attributes**. Use full path name, for example, `/sbin/ifdown`.



Type: up

Script: /usr/sbin route add -net 192.168.0.1

Figure 5.8. Defining interface scripts

Step 5. To execute multiple commands, repeat Steps 2-4.

Step 6. To set the order of commands, use the arrow buttons below the list of options.

Tip
If complex scripts have to be used, create a script file on the Zorp host using a text editor, and add an option to run it when needed.

5.1.6.2. Interface groups

To simplify the management of services that are available from multiple zones and interfaces, interfaces can be grouped. Interfaces belonging to an interface group can be controlled together: the `ifup` and `ifdown` commands support interface groups too. Firewall rules can accept connections on interface groups too. For details on zones, services, and firewall rules, see *Chapter 6, Managing network traffic with Zorp (p. 87)*.

5.1.6.2.1. Procedure – Creating interface groups

Purpose:

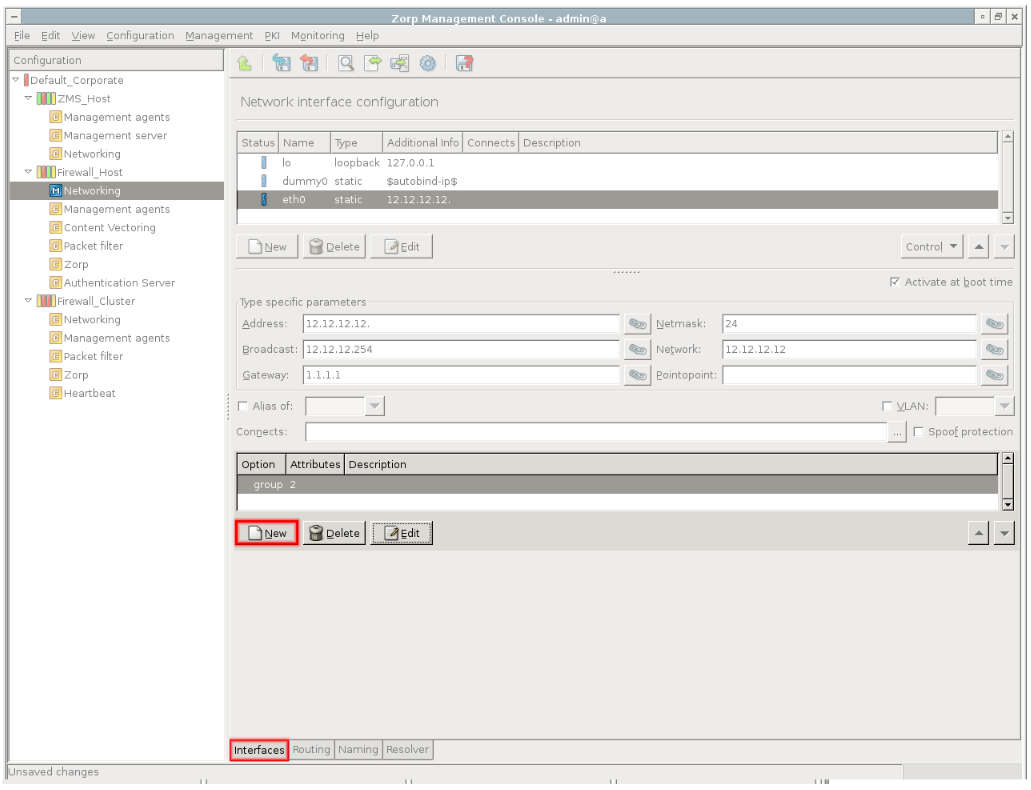
To assign interfaces to an interface group, complete the following steps.

Steps:

Step 1. Navigate to the **Networking** component and select the **Interfaces** tab.

Step 2. From the list of interfaces, select the interface to be added to the group.

Step 3. Under the list of options, click **New**.



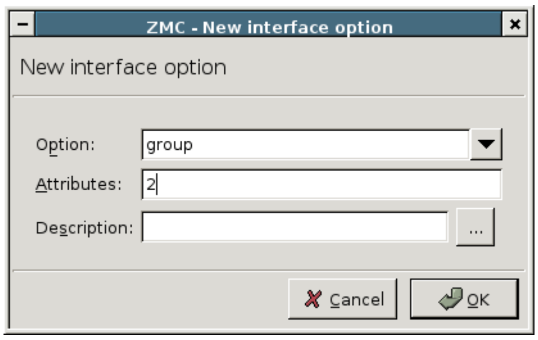
Status	Name	Type	Additional Info	Connects	Description
	lo	loopback	127.0.0.1		
	dummy0	static	$autobind-ip$		
	eth0	static	12.12.12.12.		

Address:	12.12.12.12.	Netmask:	24
Broadcast:	12.12.12.254	Network:	12.12.12.12
Gateway:	1.1.1.1	Pointopoint:	

Option	Attributes	Description
group	2	

Figure 5.9. Creating new interface groups

Step 4. From the **Option** field, select `group`.



Option:	group
Attributes:	2
Description:	

Figure 5.10. Creating interface groups

Step 5. Groups are identified by a number between *1–255*. To assign the interface to a group, enter a number into the **Attribute** field.

Step 6. Click **Ok**.

Step 7. Repeat Steps 2–5 to add other interfaces to the group.

Step 8. Commit and upload the changes. To activate the changes, restart the interface.

5.1.6.3. Other interface options

The following miscellaneous options can be configured for a network interface.

Note
It is rarely required to use these options in common scenarios. Modify these settings only if you are completely aware of all the necessary details.

- **hwaddress**: It sets the MAC address of the interface.
- **mtu**: It sets the Maximum Transfer Unit (MTU) of the interface.
- **keep-configuration**: This parameter prevents either the manually configured IPs and routes or the IPS and routes set by other processes (mostly keepalived) from being dropped at the restart of the Networking component. When this option is set to value *static*, the static addresses and routes on the actual interface are not dropped at the restart of the Networking component. Also, if this parameter is set, then after any change on these interfaces, the old values will not be removed at the restart of the Networking component, but new values will be added (for example, IP, subnet). Temporarily turning the *`keep-configuration`* parameter to *no* and restarting the node is not advised, because the networking restart will remove all settings added by other sources too. It is recommended to reboot the node after these values have been changed, or to configure these changes manually, and skip restart.

5.1.6.3.1. Procedure – Configuring interface parameters

To configure an interface parameter, complete the following steps.

Step 1. Navigate to the **Networking** component and select the **Interfaces** tab. Select the interface to be configured.

Step 2. To add a new option to the interface, click **New**.

Step 3. From the **Option** field, select the parameter that you want to configure.

Step 4. Enter the value of the parameter into the **Attributes** field.

Step 5. Commit and upload your changes. To activate the changes, restart the interface.

5.1.7. Interface status and statistics

The state of the configured interfaces is indicated by coloured leds in front of the name of the interface:

- 🟩 Green bar: the interface is up (on the host or on all cluster nodes).
  In case of Keepalived interface: the interface is up only in one cluster node.

- 🟨Yellow bar: the interface is up on some nodes, but not on all of them.
  In case of Keepalived interface: the interface is up on multiple nodes, but not on all of them.

- 🟥 Red bar: the interface is down on all nodes.
  In case of Keepalived interface: the interface is up on all nodes.

- ▮ Blue bar: Unknown.

The state of the interface is automatically updated periodically. The frequency of the update can be configured in **Edit** > **Preferences** > **General** > **Program status**.

Status update can also be requested manually from the local menu of the interface:

Step 1. Select the interface.

Step 2. Right-click on the interface and select **Refresh State.**

Note
Note that the interfaces with IPv6 addresses are always marked with blue bar.

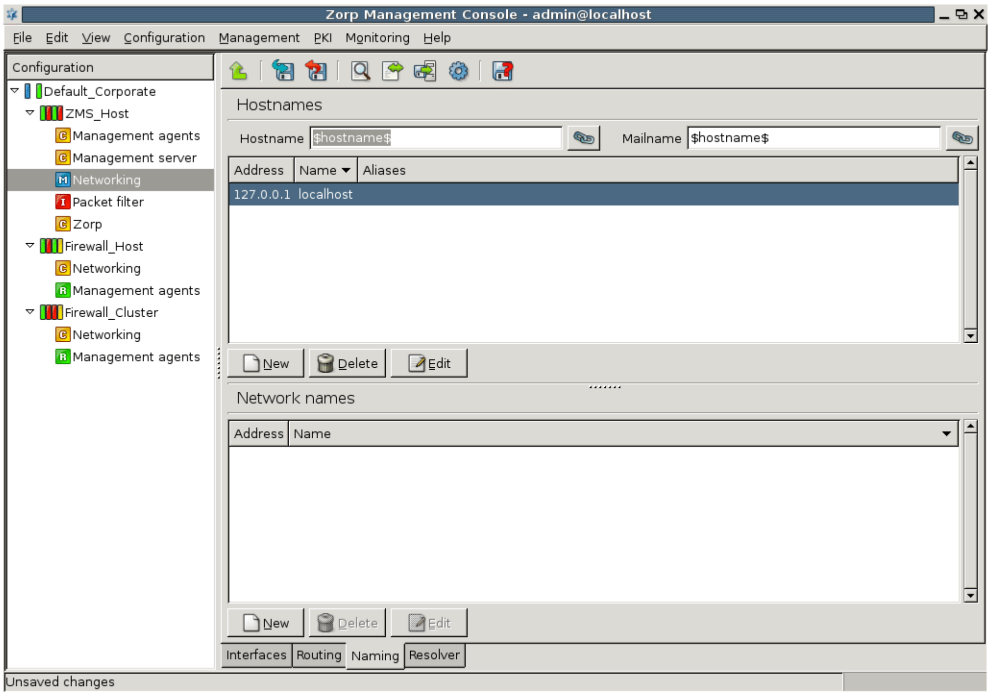
Hovering the mouse over an interface displays a tooltip with status information and detailed statistics about the interface. The following status information is displayed:

- **State**: It is the status of the interface. The possible values are: *`Up`*, *`Down`*, *`Unknown`*.
- **flags**: These are the flags applicable to the interface. The possible values are: *`UP`*, *`BROADCAST`*, *`MULTICAST`*, *`PROMISC`*, *`NOARP`*, *`ALLMULTI`*, *`LOOPBACK`*.
- **mtu**: Maximum Transmission Unit (MTU) is the maximum size of a packet (in bytes) allowed to be sent from the interface.
- **qdisc**: It is the queuing discipline. For details, see the *Traffic Control manual pages* (`man tc`).

The statistics about the traffic handled by the interface is divided into **Received** and **Sent** sections, relevant for the received/sent packets, respectively.

- **Bytes**: It is the amount of data received.
- **Packets**: It is the number of packets received.
- **Errors**: It is the number of errors encountered.
- **Dropped**: It is the number of dropped packets.
- **Overruns**: It is the number of overruns. Overruns usually occur when packets come in faster than the kernel can service the last interrupt.
- **Mcast**: It is the number of multicast packets received.
- **Bytes**: It is the amount of data sent.
- **Packets**: It is the number of packets sent.
- **Errors**: It is the number of errors encountered.
- **Dropped**: It is the number of dropped packets.
- **Carrier**: It is the number of carrier losses detected by the device driver. Carrier losses are usually the sign of physical errors on the network.
- **Collsns (Collisions)**: It is the number of collisions encountered.

5.2. Managing name resolution



Hostname: $hostname$
Mailname: $hostname$

Address	Name	Aliases
127.0.0.1	localhost	

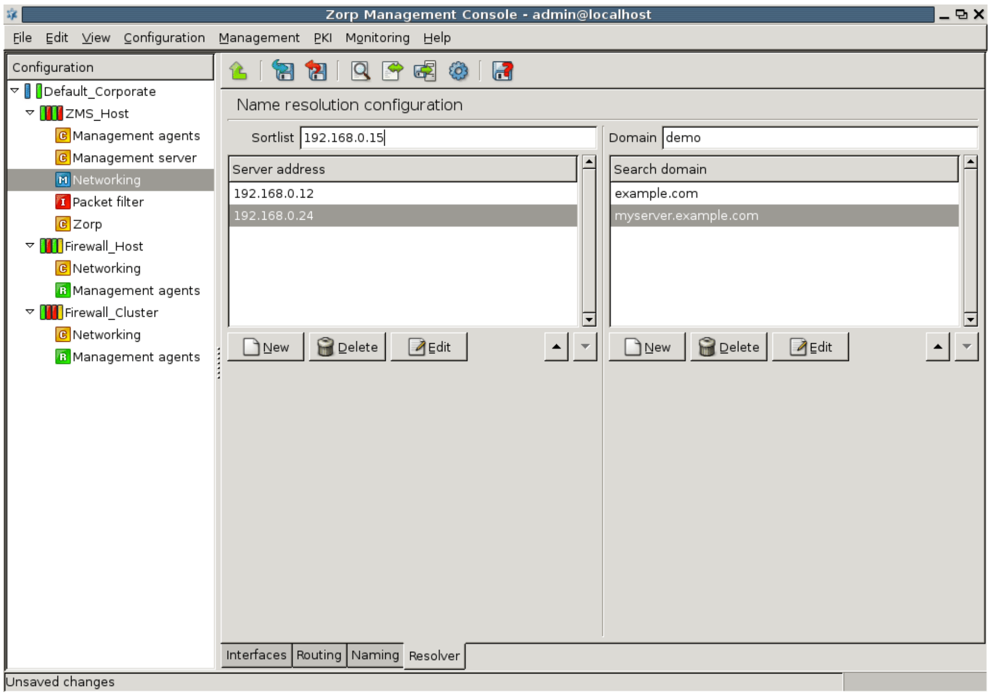
Address	Name

Figure 5.11. The Naming tab

Variables on the **Naming** `/etc/hosts` and `/etc/networks` files. Their use is mostly optional, name resolution is faster if the most important name-IP address pairs are listed in `/etc/hosts`. A correctly configured resolver can provide the same service.

Note
If you intend to use the Postfix native proxy on the Zorp host, you exceptionally have to supply the *`Mailname`* parameter.

5.3. Managing client-side name resolution



Name resolution configuration

Sortlist 192.168.0.15

Domain demo

Server address
192.168.0.12
192.168.0.24

Search domain
example.com
myserver.example.com

Figure 5.12. The Resolver tab

The client-side of DNS name resolution information can be configured on the **Resolver** tab.

Note
The Bind native proxy of Zorp cannot be configured here. For information, see *Chapter 9, Native services (p. 218)*.

In DNS terminology, the client initiating a name resolution query is called a resolver. For details about configuring a resolver correctly, refer to a good DNS documentation, see *Appendix C, Further readings (p. 485)*.

5.3.1. Procedure – Configure name resolution

Step 1. List the nameservers to be used by your host in the right pane.

Step 2. Set a priority order among the nameservers. The first one on the list is queried first.

Step 3. Set up domain search order in the left pane.
Use the buttons with triangles on the right.

This information is used when you issue a name query for a hostname but without supplying the domain name parts: for example, telnet myserver. In this case, the resolver automatically tries to append domain substitutes to the hostname in the order you specify, before sending queries to nameservers.

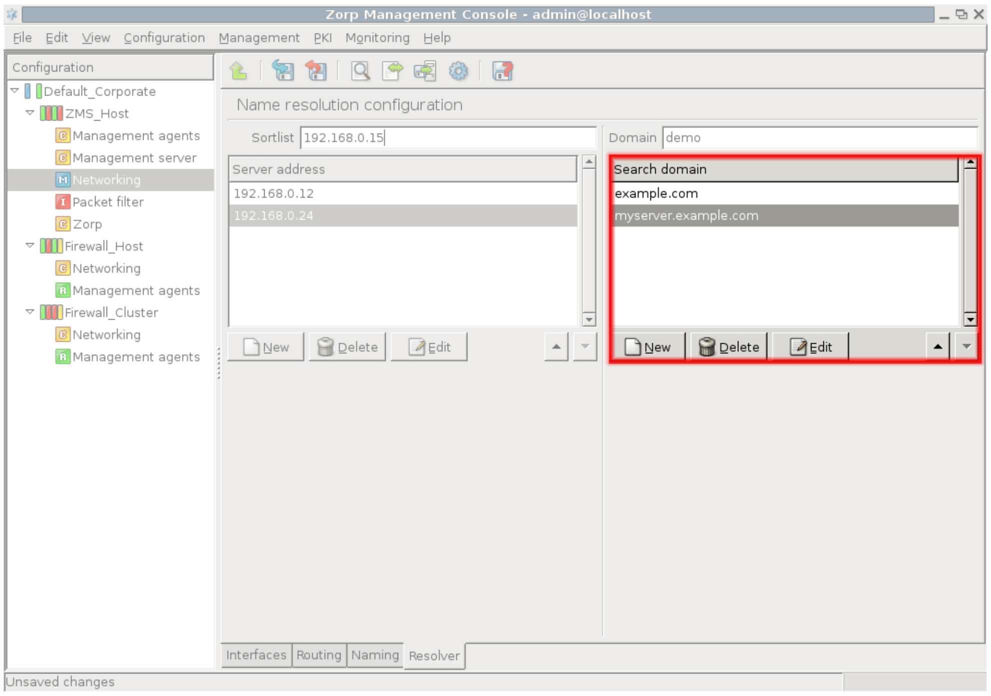


Figure 5.13. Domain search order

In the example above, this would be `example.com` and then, if the query is unsuccessful `myserver.example.com`.

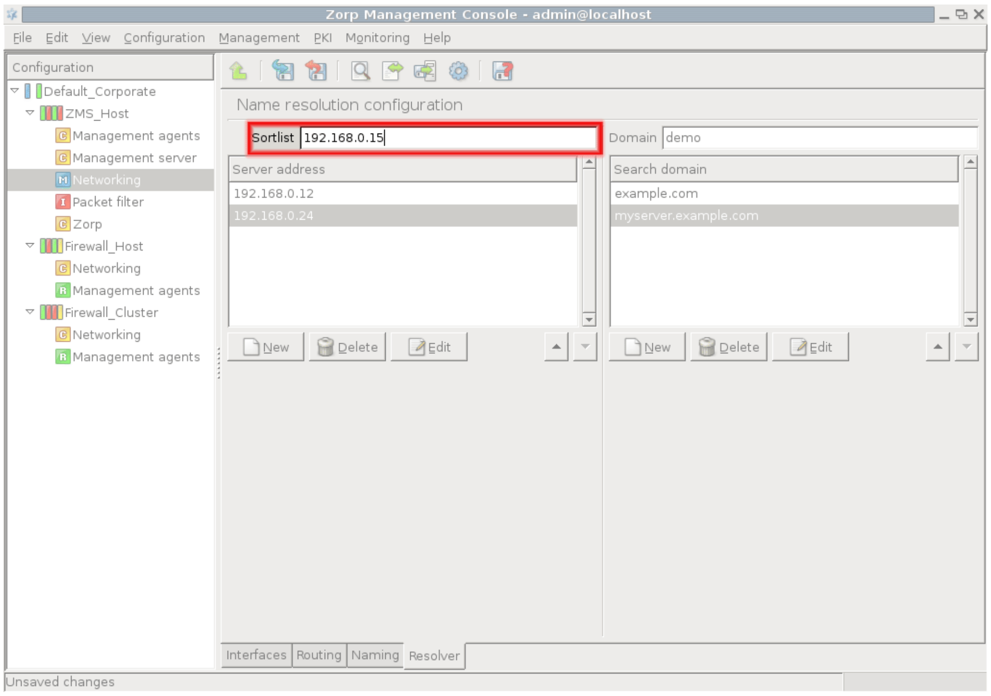
Step 4. OPTION
Define the preferred interface in the **Sortlist**.

The sortlist directive specifies the preferred interface you wish to communicate on, when, as a result of a query, you receive more than one IP addresses for a given host. The value of Sortlist can be a network IP address or a host IP address/subnet mask pair, where the subnet mask is in the classic dotted decimal format and not in CIDR notation.

Tip
The optimization using the Sortlist might be useful for firewalls with many interfaces installed, or in the following special network setup.

The firewall is connected to the Internet with two interfaces: one for a broadband, primary connection and another, lower-bandwidth backup connection through a different Internet Service Provider (ISP). If you want to reach a server on the Internet, the DNS query returns two IP addresses for the same server. From its routing table, your firewall deduces that both IP addresses are reachable, but by default it uses the IP address that was listed first in the DNS response, even if that IP address is reachable through the — slower — backup line. To avoid this situation, you can explicitly tell your resolver with the **Sortlist** feature that whenever possible, it must prefer the interface that connects to the higher-bandwidth primary line.

Note that the **Sortlist** feature provides just a preference and not an exclusive setting: if the targeted server cannot be reached via the interface designated by the sortlist parameter, the other interface(s) and IP addresses are tried.



Sortlist 192.168.0.15

Domain demo

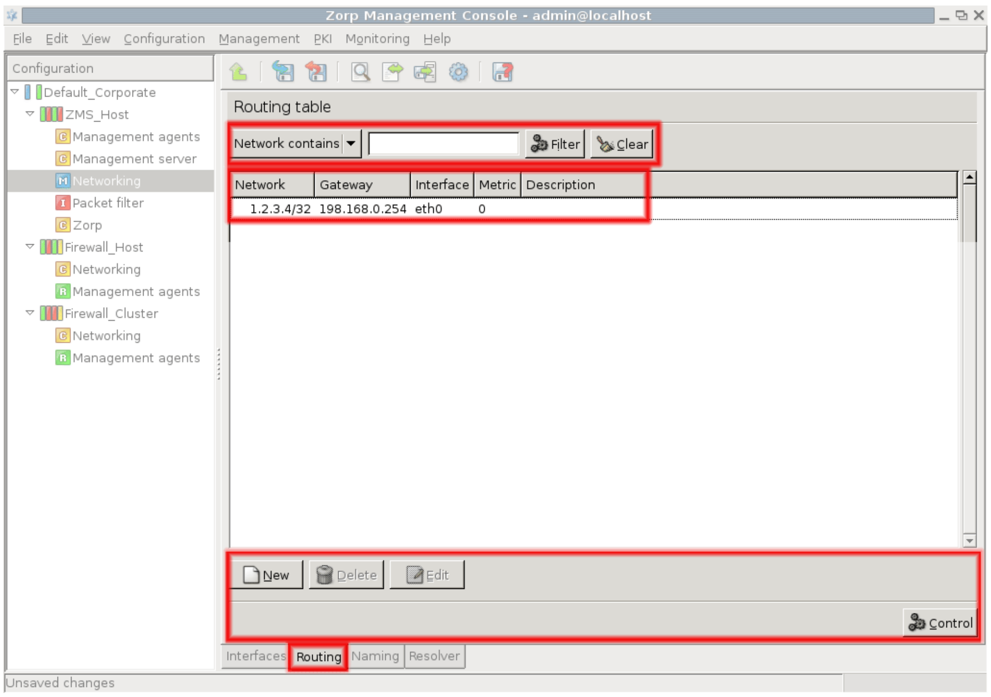
Server address
192.168.0.12
192.168.0.24

Search domain
example.com
myserver.example.com

Figure 5.14. Sortlist setting

5.4. The routing editor

If a packet has to be delivered to a remote network, Zorp consults the routing table to determine the path it should be sent to. If there is no information in the routing table, then the packet is sent to the default gateway. For maintaining and managing the routing table Zorp offers a simple, yet effective user interface on the **Routing** tab of the **Networking** ZMC component. It can be used to define static routes to specific hosts or networks through a selected interface.



Network	Gateway	Interface	Metric	Description
1.2.3.4/32	198.168.0.254	eth0	0	

Figure 5.15. The Routing tab

The **Routing** tab contains the following elements:

- a list of routing table entries in the middle of the panel
- a filter bar on the top for searching and filtering the entries
- a control bar on the bottom for managing the entries of the list and for activating the modifications of the routing table on the Zorp firewall hosts

5.4.1. Routes

A route has the following parameters:

- **Address**: It is the IP address of the network.
- **Netmask**: It is the Netmask of the network.

Note
The IP address and netmask parameters are displayed in the **Network** column of the routing entries list.

- **Gateway**: It is the IP address of the gateway to be used for transmitting the packages.
- **Interface**: It is the Ethernet interface to be used for transmitting the packages. This parameter must be selected from the combobox listing the configured interfaces.
- **Metric**: It is an optional parameter to define a metric for the route. The metric is an integer from the `0 - 32766` range.
- **Description**: These notes describe what this entry is used for.

The above parameters are interpreted the following way: messages sent to the *`Address/Netmask`* network should be delivered through *`Gateway`* using *`Interface`*.

New entries into the routing table can be added by clicking the **New** button of the control bar; existing entries can be modified by clicking **Edit**. The updated routing tables take effect only after the new configuration is uploaded to the host, and the routing tables are reloaded using the **Actions/Reload** buttons of the control bar.

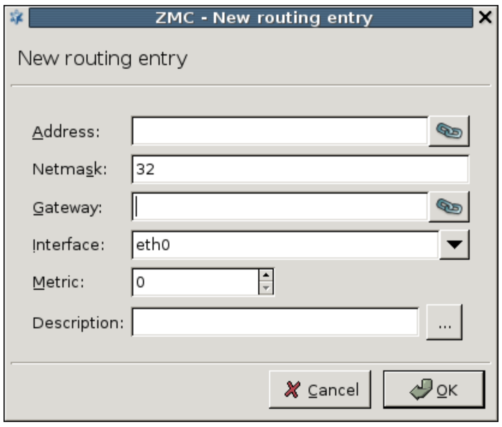


Figure 5.16. Adding new routing entries

5.4.2. Sorting, filtering, and disabling routes

The list of routing table entries can be sorted by all of its parameters (network, gateway, and so on) by clicking on the header of the respective column. By default, the list is displayed according to the general listing policy of the routing tables, that is, the networks connecting via a gateway are listed after the directly connected networks.

The list can be filtered using the filter bar above the list.

5.4.2.1. Procedure – Filtering routes

Step 1. Type the search pattern into the textbox.

Step 2. Specify the elements to be searched using the combobox on the left.

Step 3. Click **Filter**.
The list will only display the routes matching the search criteria.

Step 4. Click **Clear** to return to the full list.

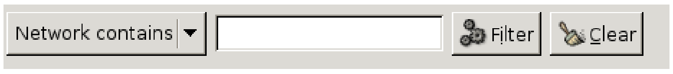


Figure 5.17. The Filter bar

Routes can be temporarily disabled by right-clicking the selected route and selecting **Disable** from the appearing local menu.

Note
The route becomes disabled only after the routing table is reloaded.

5.4.3. Managing the routing tables locally

When the Zorp host is administered via a terminal, the routes have to be entered manually by editing the `/etc/network/static-routes` configuration file. Each line of this file corresponds to a route, and has the following format:

`interface_name` to *`address/netmask`* through *`gateway`* metric *`[metric]`*

For the modifications to take effect, the routing tables have to be reloaded by issuing the `/usr/lib/zms-transfer-agent/zms-routing params="reload"` command.

Chapter 6. Managing network traffic with Zorp

This chapter describes how to allow traffic to pass the Zorp firewall. It gives detailed explanation of the many features and parameters of Zorp.

6.1. Understanding Zorp policies

This section provides an overview of how Zorp handles incoming connections, and the task and purpose of the different Zorp components.

Zorp firewall rules permit and examine connections between the source and the destination of the connection. When a client tries to connect a server, Zorp receives the connection request and finds a *firewall rule* that matches the parameters of the connection request based on the client's address, the target port, the server's address, and other parameters of the connection. The *rule* selects a service to handle the connection. The *service* determines what happens with the connection, including the following:

- *the Transport-layer protocol permitted in the traffic, for example, TCP or UDP*
- *the service started by the firewall rule.*
  This also determines the application-level protocol permitted in the traffic. Zorp uses *proxy classes* to verify the type of traffic in the connection, and to ensure that the traffic conforms to the requirements of the protocol, for example, downloading a web page must conform to the HTTP standards.
- *the address of the destination server*
  Zorp determines the IP address of the destination server using a router. *Routers* can also modify the target address if needed.
- *the content of the traffic*
  Zorp can modify protocol elements, and perform content vectoring. See *Chapter 14, Virus and content filtering using ZCV (p. 351)* for details.
- *how to connect to the server*
  For non-transparent connections, Zorp can connect to a backup server if the original is unreachable, or perform loadbalancing between server clusters.
- *who can access the service*
  Zorp can authenticate and authorize the client to verify the client's identity and privileges. See *Chapter 15, Connection authentication and authorization (p. 389)* for details.

The operations and policies configured in the service definition are performed by a Zorp *instance*.

6.2. Zones

Zones describe and map the networking environment on IP level. IP addresses are grouped into zones; the access policies of Zorp operate on these zones. Zones specify segments of the network from which traffic can be received (source), or sent to (destination), through the firewall. Zones in Zorp can contain:

- IP networks,
- subnets,
- individual IP addresses, and
- hostnames.

Zone management is handled by kzorp daemon (kzorpd). kzorpd is responsible for maintaining zone address information in kzorp kernel modules and also for updating dynamic address information in hostname-based zones.

The actual implementation of a zone hierarchy depends on the network environment, the placement and the role of Zorp firewalls, the security policy, and so on.

The Internet zone which covers all possible IP addresses is defined on every site by default. If an IP address is not included in any user-defined zones, it belongs to the Internet zone. Zorp policies permit traffic between two or more zones, so at least another zone — the intranet — must be created. Usually a special zone called demilitarized zone (DMZ) is defined for servers available from the Internet.

Zones in Zorp can have a hierarchy, with a zone containing many subzones that may have their own subzones, and so on. From these zones, a tree hierarchy can be constructed. This hierarchy is purely administrative and independent from the IP addresses defined in the zones themselves: for example, a zone that contains the *192.168.7.0/24* subnet can have a subzone with IP addresses from the *10.0.0.0/8* range.

A network can belong only to a single zone, because otherwise the position of IP addresses in the network would be ambiguous.

The zone hierarchy is independent from the subnetting practices of the company or the physical layout of the network, and can follow arbitrary logic. The zone hierarchy applies to every host of a site.

Note
Subnets can be used directly in Zorp configurations, it is not necessary to include them in a zone.

Note
It is recommended to follow the logic of the network implementation when defining zones, because this approach leads to the most flexible firewall administration. Plan and document the zone hierarchy thoroughly and keep it up-to-date. An effective and usable zone topology is essential for successful Zorp administration.

6.2.1. Managing zones with ZMC

By default, ZMC defines a zone called *internet* on every site. The *internet* contains the *0.0.0.0* and the *::0* networks with the *0* subnet mask. This zone means any network: every IP address not belonging to any other zone belongs to the *internet* zone.

Note
Zorp uses the CIDR notation for subnetting.

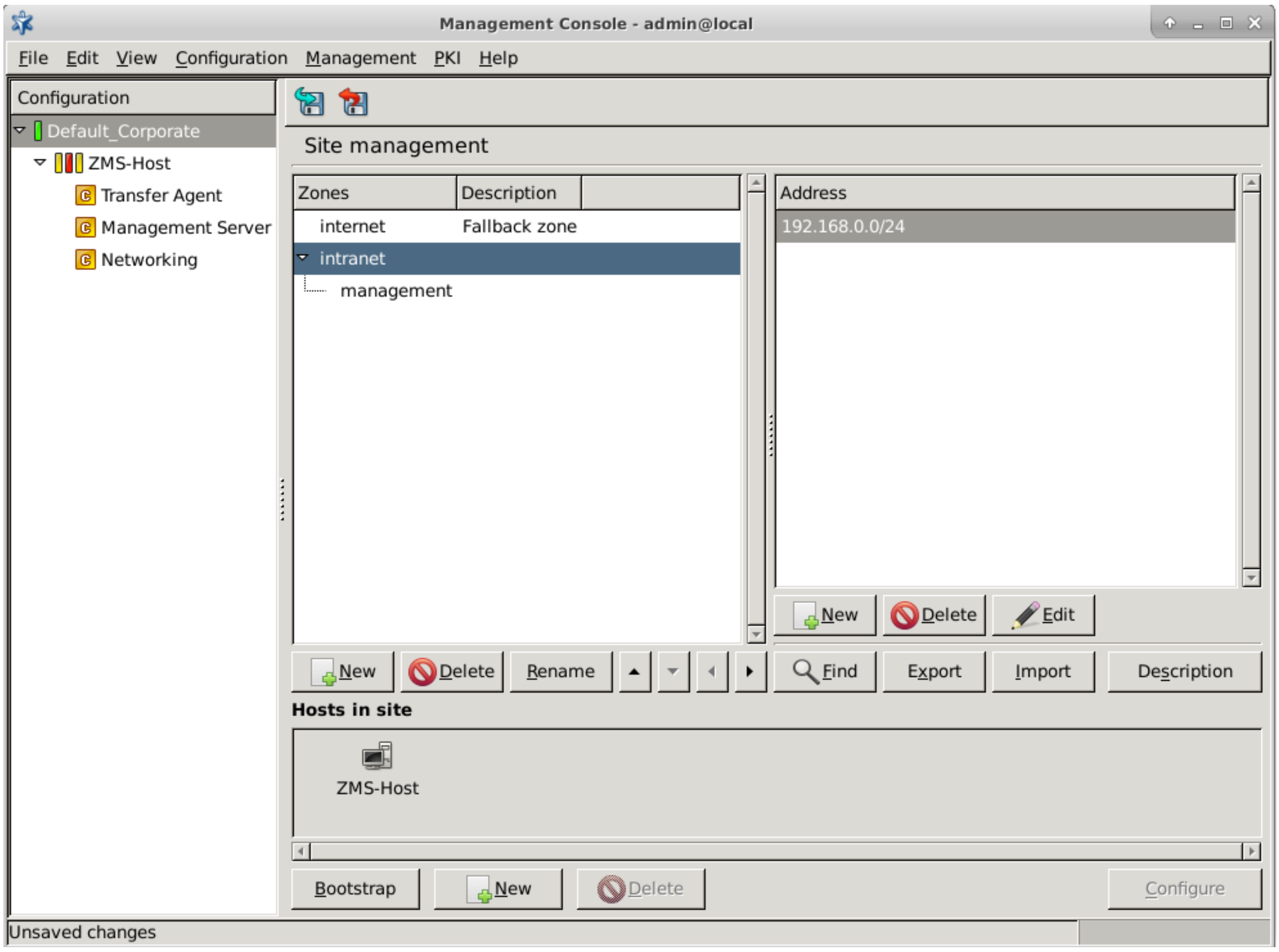


Figure 6.1. Zones

The *`internet`* zone is typically used in firewall rules where one side of the connection cannot be defined more exactly.

Example 6.1. Using the Internet zone
The Internet zone identifies all external networks. To allow the internal users to visit all web pages, simply set the destination zone of the HTTP service to *`Internet`*. For details on creating services, see *Section 6.4, Zorp services (p. 115)*.

Zones are managed on the **Site** component in ZMC. The left side of the main workspace displays the zones defined on the site and their descriptions. IP networks that belong to the selected zone are displayed on the right side of the workspace.

Note
The **Zorp** ZMC component has a shortcut in its icon bar to the zone editor. The zone hierarchy applies to all firewalls of the site, therefore carefully consider every modification and its possible side-effects.

Use the control buttons to create, delete, or edit the zone definitions and the IP networks. Use the arrow icons to organize the zones into a hierarchy (see *Section 6.2.3, Zone hierarchies (p. 93)* for details).

If a zone is created, modified or deleted in a ZMC, the change is immediately visible in the zone lists of the same ZMC without committing the changes. If these changes to a zone or zones are committed, the changes become visible in the zone information of other ZMCs as well.

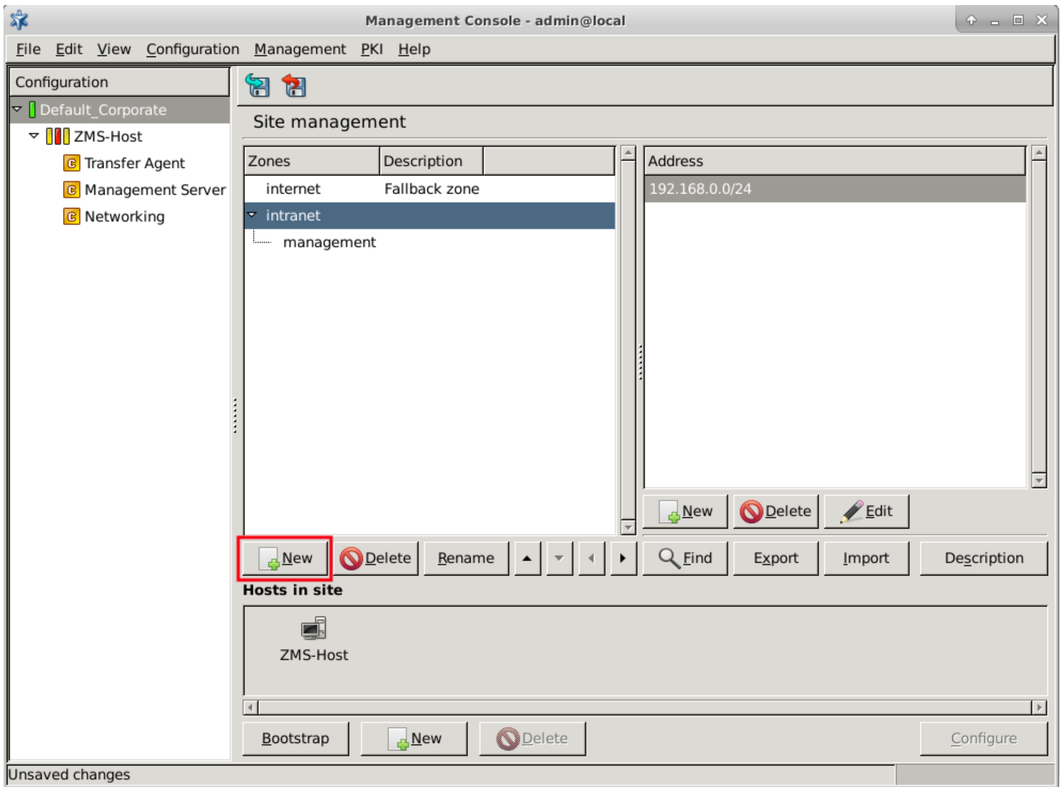
Example 6.2. Subnetting
Suppose you have the following IP address range to put into a zone: `1.2.50.0` – `1.2.70.255`. You can either define 21 IP subnets with `/24` mask or you can define six subnets in the following manner: `1.2.50.0/23`, `1.2.52.0/22`, `1.2.56.0/21`, `1.2.64.0/22`, `1.2.68.0/23`, `1.2.70.0/24`. Whether you have a switched/routed network or you actually use `/24` subnets is irrelevant from the zone's (Zorp's) point of view. As long as it encounters an IP address from the range `1.2.50.0` – `1.2.70.255`, it will consider it a member of the given zone.

Furthermore, if you define Zone A with the IP network `10.0.0.0/8` and Zone B consisting of the network `10.0.1.0/24` and the machine, Computer C with the IP address of `10.0.1.100/32`, from an IP addressing point of view, Computer C belongs to both subnets, but the Zorp rule applied in this and similar cases is, that the machine is always considered to belong to the more specific network (and thus the zone), as also specified by the CIDR method. In this example it is Zone B.

6.2.2. Procedure – Creating new zones

To create a new zone on the site, complete the following steps.

Step 1. Select the site from the configuration tree and click **New**.



Zones	Description
internet	Fallback zone
intranet	
management	

Address
192.168.0.0/24

Figure 6.2. Creating a new zone

Step 2. Enter a name for the zone in the displayed window.

Tip
Use descriptive names and a consistent naming convention. Zone names may refer to the physical location of the network or the department using the zone (for example, `building_B`, or `marketing`).

Step 3.

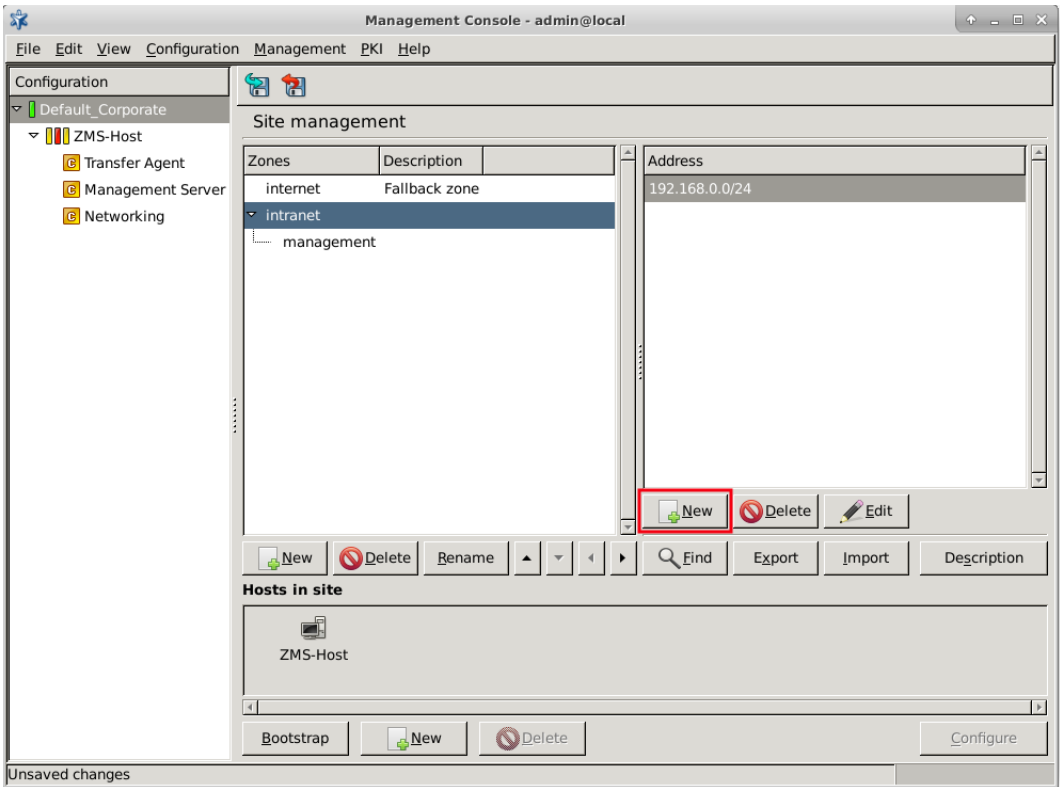


Figure 6.3. Creating a new network in a zone

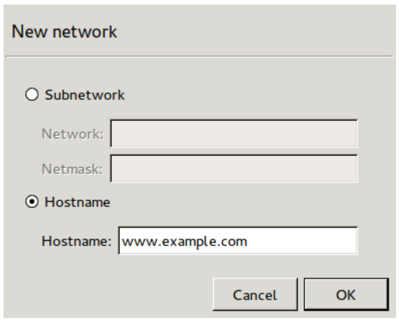
To add an IP network to the zone, click **New** in the Networks pane.

Step 4.

- To add a network or an IP address to the zone, select **Subnetwork**, fill the **Network** and **Netmask** fields, then click **Ok**.
- To add a hostname to the zone, select **Hostname**, enter the hostname into the **Address** field, then click **Ok**. For details on using hostnames in zones, see *Section 6.2.4, Using hostnames in zones (p. 96)*.

Repeat this step to add other networks to the zone.

Note
The new zone has effect only if used in a firewall rule definition.


New network

( ) Subnetwork

Network:

Netmask:

(●) Hostname

Hostname: www.example.com

Figure 6.4. Adding networks to a zone

6.2.3. Zone hierarchies

Zones can be organized into a tree, much like the directories of a file system. Define a topmost zone and with many subzones, each for administratively different parts of your networks. A zone and its subzone have parent-child relationship: child zones automatically inherit all properties and settings of their parents. For example, Zone A is the parent zone of Zone B, and all clients in Zone A may browse the web through HTTP. Zone B inherits this setting, so all clients of Zone B have unrestricted HTTP access.

To stop a zone from inheriting the properties of the parent zone, use a DenyService. For details on DenyServices, see *Procedure 6.4.3, Creating a new DenyService (p. 119)*.

Zones can be reorganized as needed.

Note
Changing parent-child relations also changes the inheritance chain — which might cause unexpected results on your firewall policies. Make sure to keep up-to-date documentation of your firewall configuration.

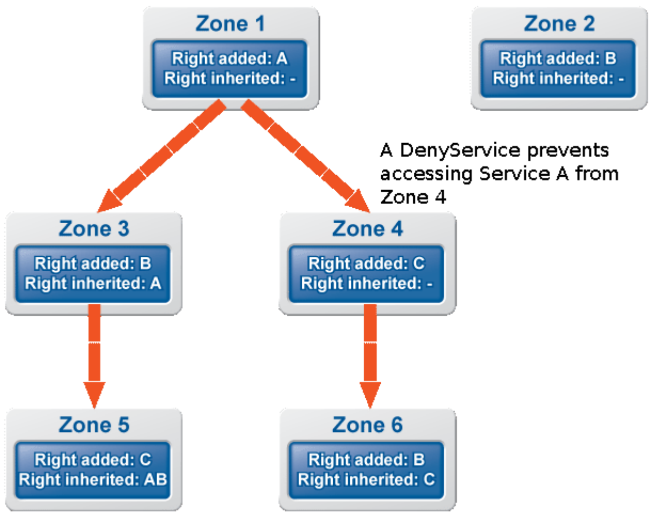


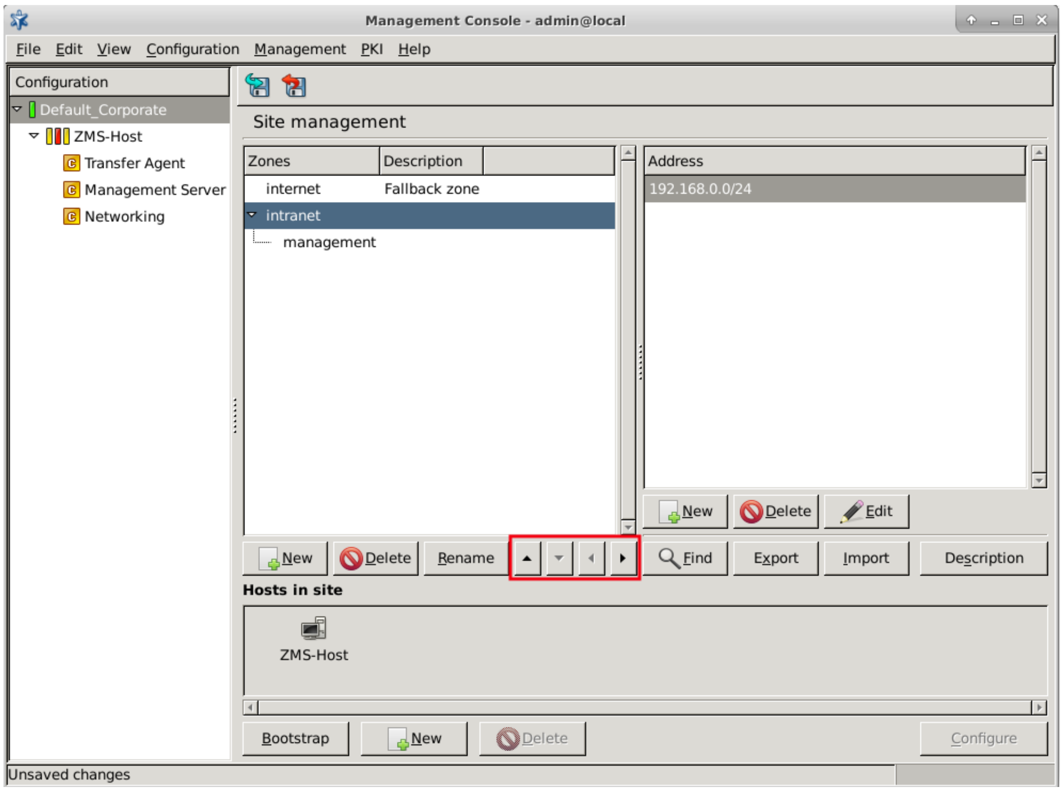
Figure 6.5. Zones, inheritance, and DenyServices

6.2.3.1. Procedure – Organizing zones into a hierarchy

To organize zones into a hierarchy, complete the following steps.

Step 1. Select the site from the configuration tree in ZMC.

Step 2. Move the child zone below its parent by using the up and down arrows located next to the **Find** button.



Zones	Description
internet	Fallback zone
intranet	
management	

Address
192.168.0.0/24

Figure 6.6. Configuring zone hierarchy

Step 3. Click the right arrow to make the selected zone the child of the zone above it.

Step 4. **Commit** the changes to the site.

Note
Zone definitions are site-wide, so modifications are effective on every firewall of the site.

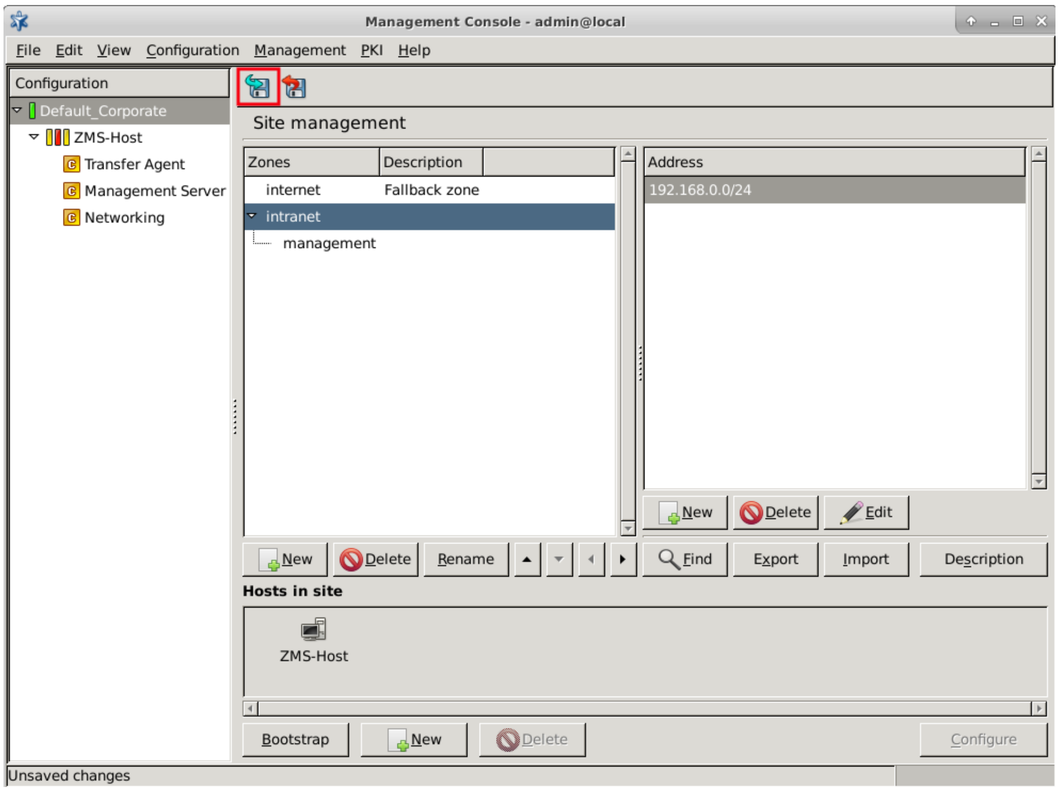


Figure 6.7. Committing changes to the site

Step 5. Select all hosts of the site and upload the configuration.
This step is required because changes in the zone hierarchy must be uploaded to all firewall nodes.

Step 6. Select all hosts of the site, click the **Control** icon of the icon bar and reload the configuration.

To remove a child zone from the hierarchy, select the zone and click the left arrow.

6.2.4. Using hostnames in zones

Starting with Zorp 5.0, you can directly use hostnames in zones. During startup, Zorp automatically resolves these hostnames to `/32` IP addresses, and updates them periodically to follow any changes in the IP addresses related to the hostname. When using hostnames in zones, note the following considerations and warnings:

- Ensure that your Domain Name Server (DNS) is reliable and continuously available. If you cannot depend on your DNS to resolve the hostnames, do not use hostnames in zones.
- Do not use zones that include hostnames to deny access, that is, do not use such zones in DenyServices. If Zorp cannot resolve a hostname, it will omit the hostname from the zone. If the zone contains only a single hostname (because you want to use it to restrict access to a specific site), the zone will be empty, that will never match any connection. If you have a firewall rule that is more permissive than the DenyService you are using the zone with the hosname, this more permissive rule will be effective, permitting traffic you want to block. (For example, you create a rule that permits HTTP traffic to the Internet, and a DenyService to block HTTP traffic to the example.com hostname. If Zorp cannot

resolve the example.com hostname, then the broader, more permissive rule will permit traffic to the example.com site.)

- kzorp, besides maintaining zone address information in kzorp kernel modules, also enables the filtering and blocking of any, possibly illegitimate, so called 'bogus' IP addresses.
  The filtering of the DNS-based zone IP addresses is from now on set by default in the configuration. The level of filtering is set to the recommended value, 3 by default, which indicates the following level of filtering:

Filtering level	Filtering
0	No filtering takes place.
1	Filtering of invalid host addresses takes place: unspecified addresses (`0.0.0.0/32`, `::/128`).
2	Filtering loopback address ranges takes place (`127.0.0.0/8`, `::1/128`).
3	Filtering of private address ranges (`192.168.0.0/16`, `10.0.0.0/8`, `172.16.0.0/12`, `fc00::/7`), link-local address ranges (`169.254.0.0/16`, `fe80::/10`) and multicast ranges (`224.0.0.0/4`, `ff00::/8`) takes place.

  Table 6.1. Filtering levels

  Note, that although up until now, the kzorpd configuration options could only be changed via the command line interface, now it is already possible to make changes to the actual kzorpd configuration file with the help of a text editor.

  If the level of filtering is requested to be configured differently than the recommended value, it is possible to change it in the kzorpd configuration file with the help of text editor.

  For details see the kzorpd and the kzorpd configuration file manual pages in Appendix C, Zorp manual pages in *Zorp Professional 7 Reference Guide*.

- If the hostname is resolved to an IP address that is explicitly used in another zone, then Zorp will use the rule with the explicit IP address. For example, you have a zone that includes the example.com hostname, another zone that includes the `192.168.100.1/32` IP address, and you have two different rules that use these zones (Rule_1 uses the hostname, Rule_2 the explicit IP address). If the example.com hostname is resolved to the 192.168.100.1 IP address, Zorp will use Rule_2 instead of Rule_1.
- If more than one hostname is resolved to the same IP address, Zorp ignores that specific IP address associated with more hostnames. Consequently, it is not possible to use a hostname in a zone if the server uses name-based virtual hosting.
- Zones are global in Zorp, and apply to all firewalls of the site, so carefully consider every modification of a zone, and its possible side-effects.

6.2.5. Finding zones

To find a zone or a subnet, select the site in the configuration tree and click the **Find** button.

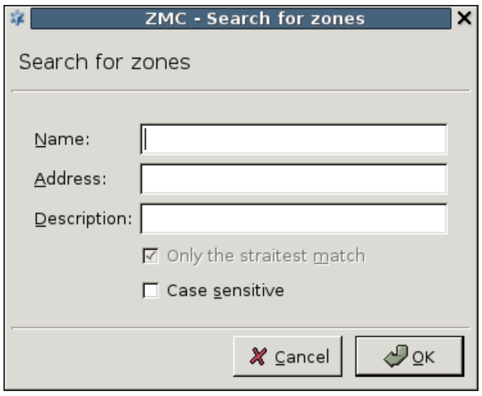


Figure 6.8. Finding zones and subnets

You can search for the name of the zone, or for the IP network it contains. When searching for IP networks, only the most specific zone containing the searched IP is returned. If an IP address belongs to two different zones, the straightest match returns the most specific zone.

Example 6.3. Finding IP networks
Suppose there are three zones configured: *Zone_A* containing the *10.0.0.0/8* network, *Zone_B* containing the *10.0.0.0/16* network, and *Zone_C* containing the *10.0.0.25* IP address. Searching for the *10.0.44.0* network returns *Zone_B*, because that is the most specific zone matching the searched IP address. Similarly, searching for *10.0.0.25* returns only *Zone_C*.

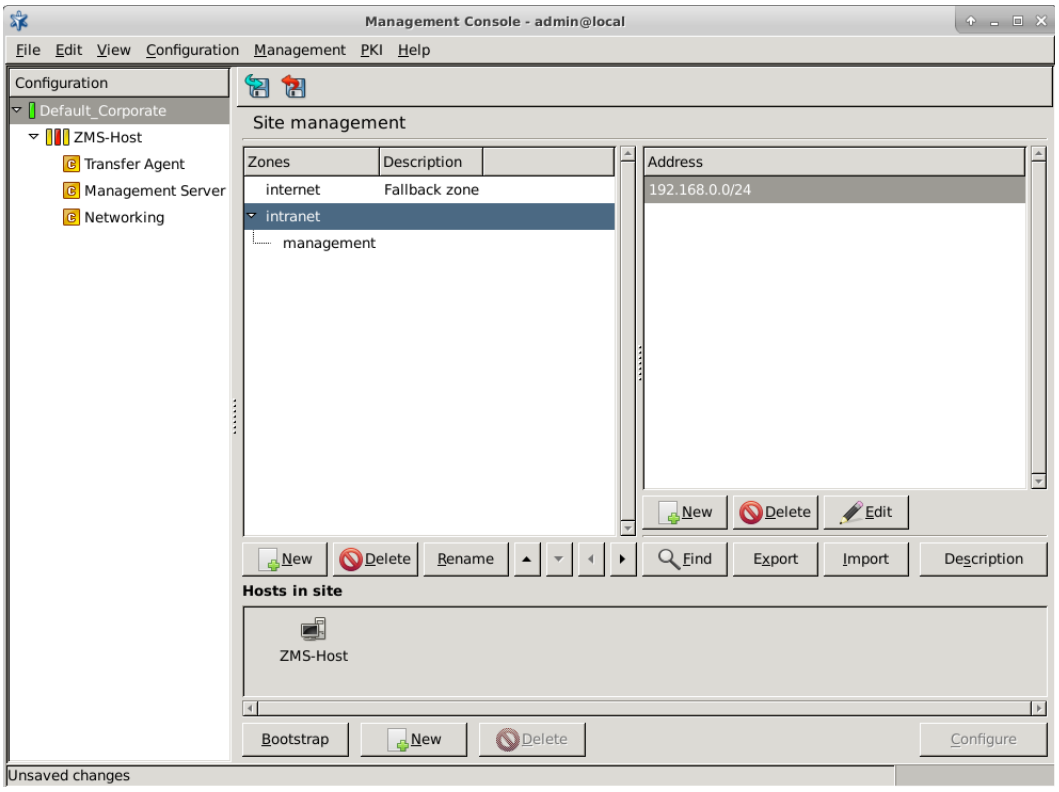
This approach is used in the service definitions as well: when a client sends a connection request, Zorp looks for the most specific zone containing the IP address of the client. Suppose that the clients in *Zone_A* are allowed to use HTTP. If a client with IP *10.0.0.50* (thus belonging to *Zone_B*) can only use HTTP if *Zone_B* is the child of *Zone_A*, or if a service definition explicitly permits *Zone_B* to use HTTP.

Tip
The **Find** tool is especially useful in large-scale deployments with complex zone and subnet structure.

6.2.6. Procedure – Exporting zones

Follow the steps to export the available zones and the zone-related pieces of information.

Step 1. Select the **Export** button to export the available zones and the zone-related pieces of information.
There is no need to select the zone or zones for export. All information on all zones will be exported.



Zones	Description
internet	Fallback zone
intranet	
management	

Address
192.168.0.0/24

Figure 6.9. Exporting zones and zone-related pieces of information

Step 2. Name the file to be exported, add the .csv extension to the file and save it.

Note
Make sure that the .csv extension is added to the exported file, otherwise the file will not be listed as an available file for any other activity, due to the missing file format.

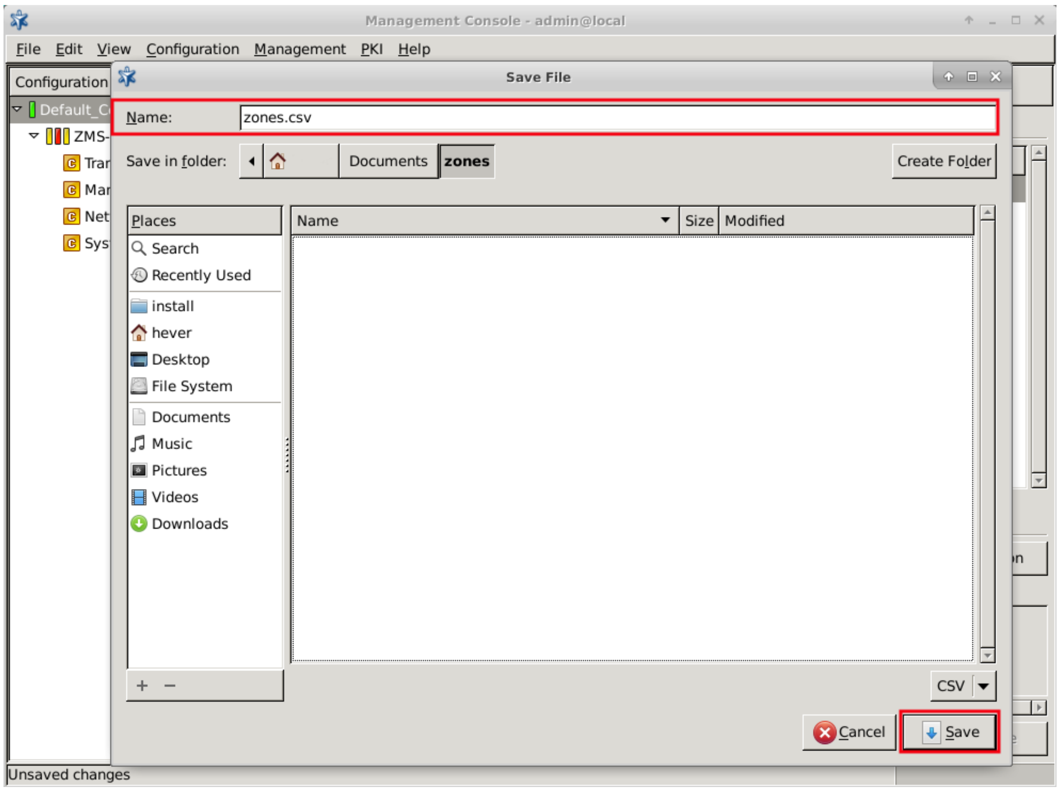


Figure 6.10. Naming the file to be exported

The following zone-related pieces of information are exported:

- Name: It is the name of the actual zone.
- Parent: It defines the parent for the actual zone.
- Subnet: It identifies the subnets for the zone in a comma-separated list.
- Host: It defines the hosts for the zone in a comma-separated list.
- Description: It is possible to provide a description for the zone.

The zone-related pieces of information are received in .csv format.

6.2.7. Procedure – Importing zones

Follow the steps and pay attention to the related considerations listed below to import zones.

Step 1. Select the **Import** button to import zones.

Note
Note: Only files with .csv extension are displayed for the activity.

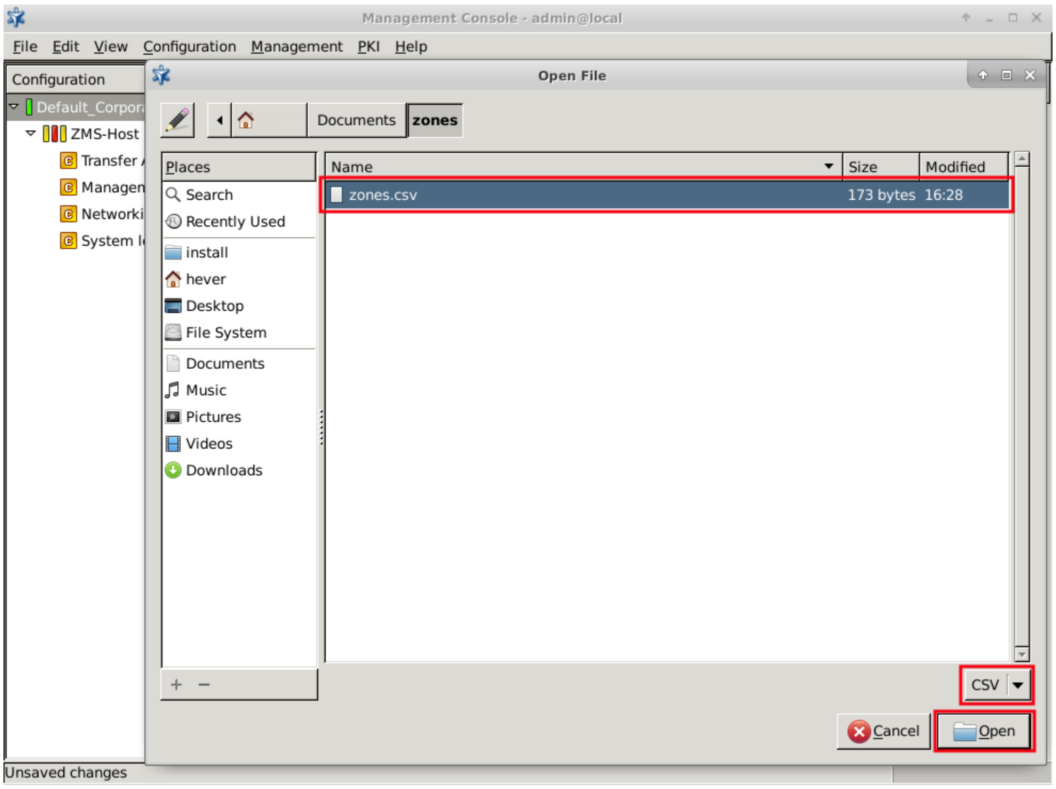
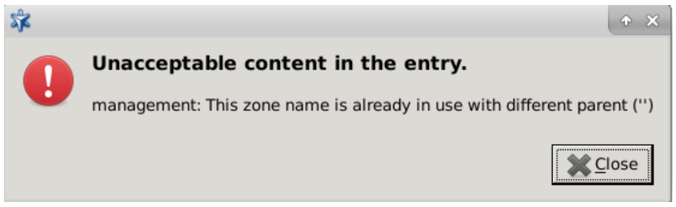


Figure 6.11. Selecting the zone for import

Step 2. Select the file to be imported and click **Open**.
Consider the followings for importing zones:

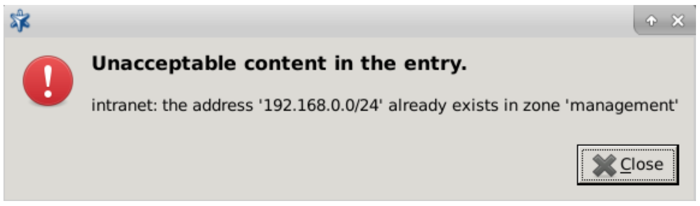
- A zone is identified by two parameters, namely its name and its parent.
- If the zone exists already, those zone-related host and subnet information will be imported from the file which do not belong to any zones in ZMC yet. That is, if the file contains hosts and subnets that exist already in ZMC and belong to that specific zone, the import process proceeds. The import process is aborted however, if the hosts and subnets belong already to another zone in ZMC.
- If the zone to be imported does not exist yet, all zone parameters are imported.
- Invalid zones are not imported and no warning is provided on that. However the import processes of invalid zones are aborted with a warning information on the abortion of the process.
  - If the zone selected for import has the same name as an existing zone in the ZMC, but the parent is different, the import process will be aborted as it is considered to be a different zone then.


Unacceptable content in the entry.

management: This zone name is already in use with different parent ('')

Figure 6.12. Zone with the same name but with different parent

- Also, the zone is considered invalid, if its subnet is already in use by an existing zone. The process will be aborted.


Unacceptable content in the entry.

intranet: the address '192.168.0.0/24' already exists in zone 'management'

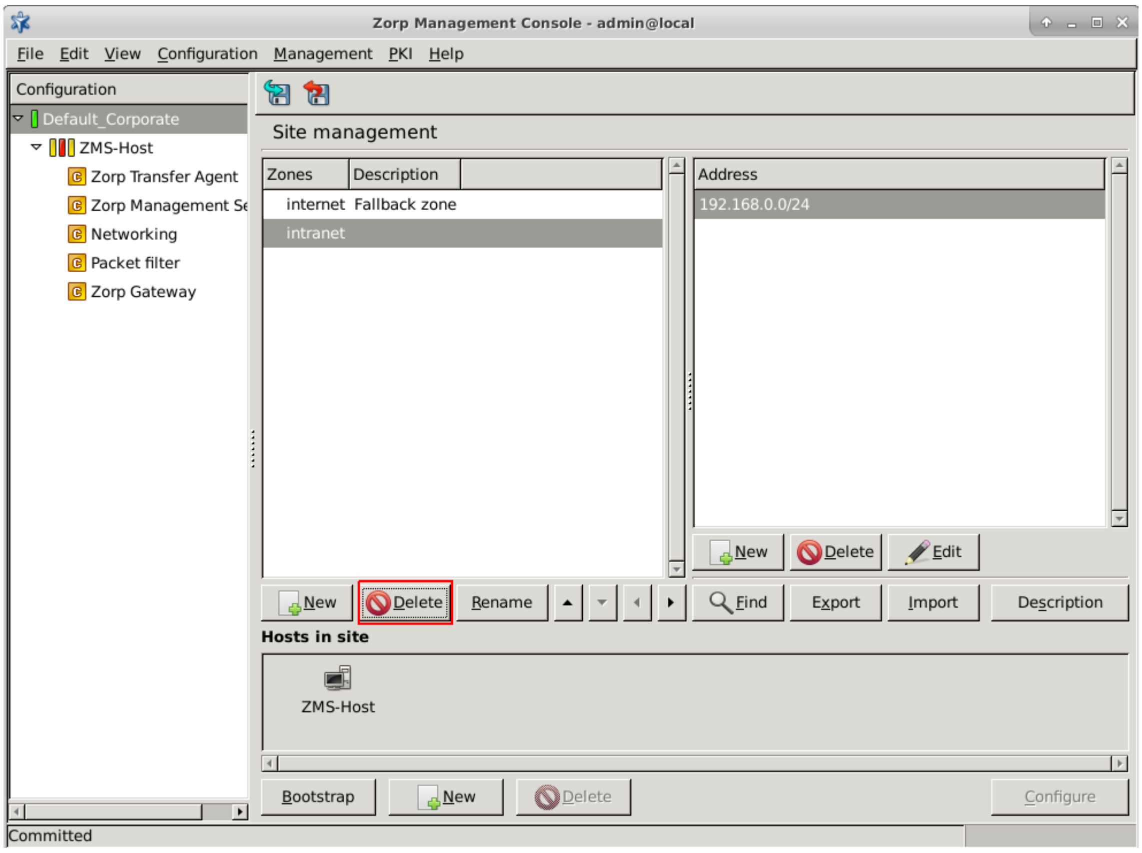
Figure 6.13. Zone with the same subnet

6.2.8. Procedure – Deleting a zone or more zones simultaneously

Follow the steps below in order to delete a zone, or even multiple zones at the same time. The deletion of multiple zones takes place one by one.

Step 1. Select the site from the configuration tree.

Step 2. Select the zone or multiple zones for deletion and click **Delete**.

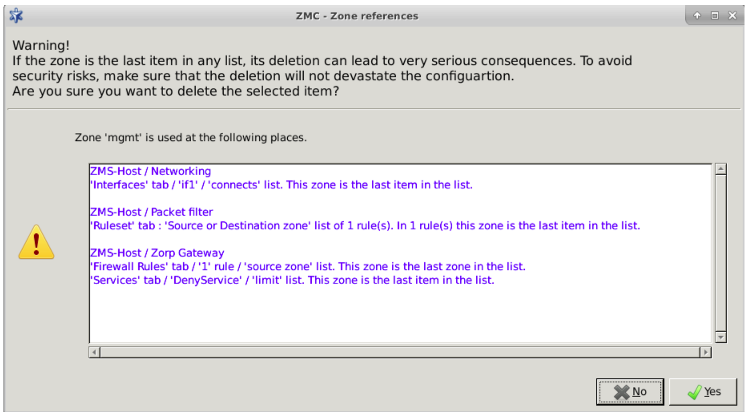


Zones	Description
internet	Fallback zone
intranet	

Address
192.168.0.0/24

Figure 6.14. Deleting a zone or multiple zones

If a zone is not referenced elsewhere in the configuration, it is deleted without any notification. If a zone, selected for deletion is referenced somewhere in the configuration, a warning pops up requesting confirmation on the deletion. If multiple zones are selected for deletion, the warning -if applicable- appears for each zone one by one:



Warning!
If the zone is the last item in any list, its deletion can lead to very serious consequences. To avoid security risks, make sure that the deletion will not devastate the configuartion.
Are you sure you want to delete the selected item?

Zone 'mgmt' is used at the following places.

ZMS-Host / Networking
'Interfaces' tab / 'if1' / 'connects' list. This zone is the last item in the list.

ZMS-Host / Packet filter
'Ruleset' tab : 'Source or Destination zone' list of 1 rule(s). In 1 rule(s) this zone is the last item in the list.

ZMS-Host / Zorp Gateway
'Firewall Rules' tab / '1' rule / 'source zone' list. This zone is the last zone in the list.
'Services' tab / 'DenyService' / 'limit' list. This zone is the last item in the list.

Figure 6.15. Warning on the deletion of a zone

It is specifically listed in the dialogue window, where the actual zone is referenced in the configuration.

It is possible at this stage not to delete the zone for which the warning appears. In this case, the administrator shall select **No** in the *Warning on the deletion of a zone* window, consequently the zone will not be deleted.

The administrator can search for the zone references in the configuration, based on the list of references displayed in the dialogue window, and make changes to their configuration, if necessary. The references can, for example, be deleted, or the referenced zones can be exchanged with other zones. If these corrections are not completed in the configuration, and a zone, that is referenced in the configuration, is deleted, than the reference to this zone in the configuration will also be deleted together with deletion of the zone. In such cases it is crucial to pay attention to these configuration details, so that the configuration is not semantically ruined. For example, in case a source or destination list of a firewall rule is emptied by the deletion of its last component, that is a zone, then the configuration might end up with a rule that matches all, or on the contrary, we might create a rule that does not let pass anything.

6.3. Zorp instances

6.3.1. Understanding Zorp instances

Instances of Zorp are groups of services that can be controlled together. A Zorp firewall can run multiple instances of the Zorp process. The main benefits of multiple firewall instances are the following:

- Administration
  A typical firewall handles many types of traffic, many different protocols. These protocols might have different administrative requirements. Inbound traffic is usually handled differently from outbound traffic. For these reasons, using multiple firewall instances can make administration more transparent.

- Availability
  If an error (for example, misconfiguration) occurs and the firewall instance stops responding, no traffic can pass the firewall. However, an error usually affects a single instance; the other ones are still functional, so only the traffic handled by the crashed instance stops. Instances can be controlled (started, restarted, stopped) individually from each other. This is important, because stopping or restarting an instance stops all traffic handled by the instance.

  Consider the following example. A firewall uses two instances: *`Instance_A`* for all e-mail related traffic (the POP3, IMAP, and the SMTP protocols) and *`Instance_B`* for everything else (HTTP, and so on). If *`Instance_A`* stops because of an error, or is stopped by the administrator, no e-mails can be sent or received. However, all other network traffic is working.

- Performance
  Separate firewall instances have separate processes and separate sets of threads, significantly increasing the performance on multiprocessor systems (SMP or HyperThreading).

- Logging
  Log settings are effective on the instance level; different instances can log in differently. For example, if logging level higher than the average is required for a type of traffic, it might worth to create an instance for this traffic and customize logging only for this instance.

Note
Although creating instances is beneficial, the number of instances that can run on a system is limited.

Each instance is a separate process and requires its own space in the system memory — quickly consuming the limited physical resources of the computer. More instances do not necessarily make configuration tasks easier, and complex configuration increases the chance of human errors.

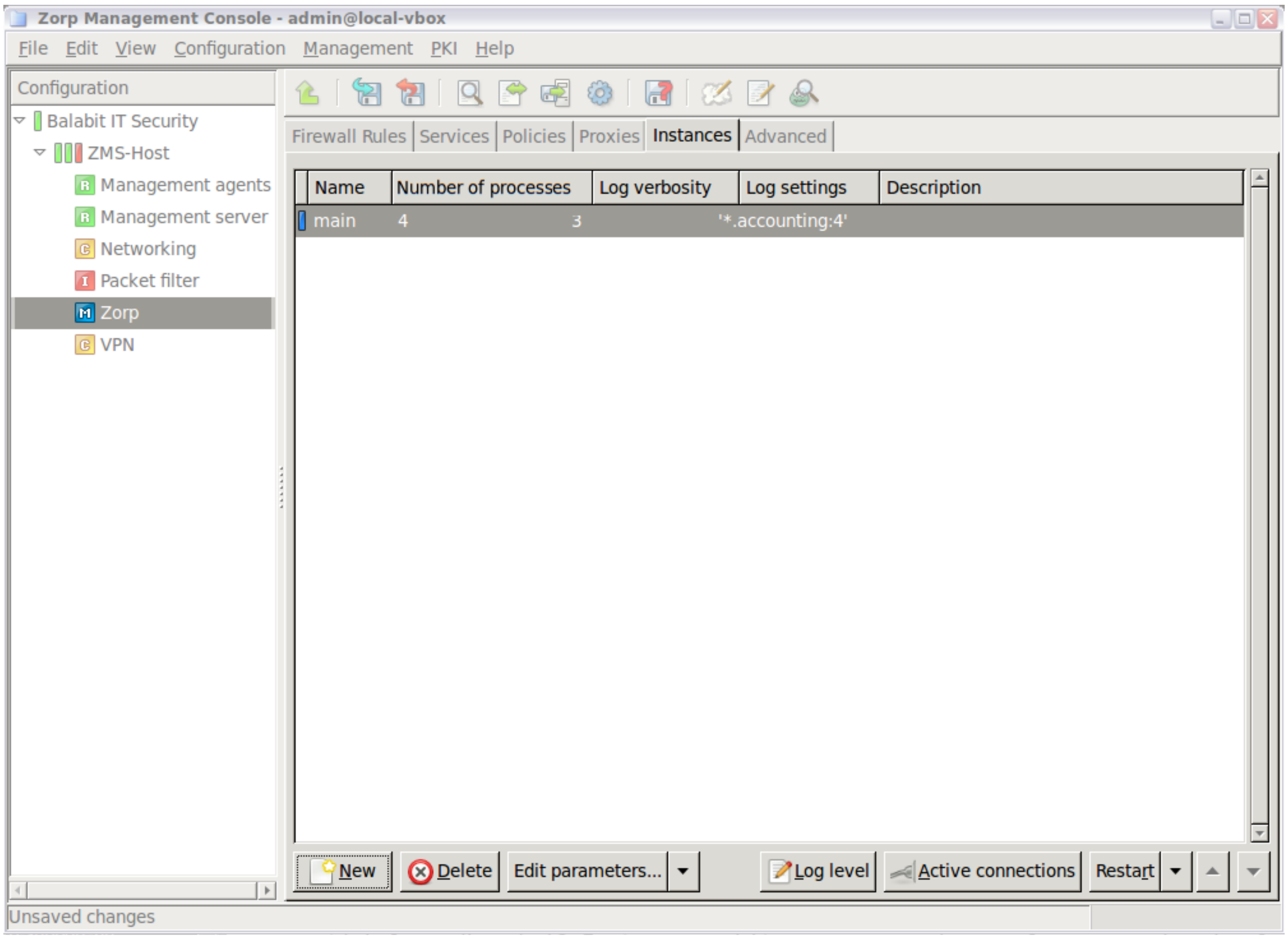
Keep instance number relatively low unless you have a solid reason to use many instances.

Instances usually handle traffic based on the protocol used by the traffic, the direction of the traffic, or a special characteristic of the traffic (for example, requires authentication). It is common practice to define an instance for all inbound traffic that handles all services accessible from the Internet, and another one for all traffic that the clients of intranet are allowed to use. Consider creating a separate instance for:

- special services, for example, mission-critical traffic;
- traffic accessing critical locations, for example, your servers;
- traffic that requires outband authentication.

6.3.2. Managing Zorp instances

To manage Zorp instances, navigate to the **Instances** tab of the **Zorp** ZMC component.



Name	Number of processes	Log verbosity	Log settings	Description
main	4	3	'*.accounting:4'	

Figure 6.16. Managing instances

The following information is displayed for each instance:

- **Name**: the name and state of the instance
  The colored led shows the state of the instance:
  - Green – Running
  - Red – Stopped
  - Blue – Unknown
    New instances that have not been started yet are in this state.
- **Number of processes**: the number of CPU cores that the instance can maximally use
  Select **Edit parameters** > **General** > **Number of processes** to modify this setting. The default value is: 1.
- **Log verbosity**: the log level set for the instance
- **Log settings**: the log specifications of the instance
- **Description**: a description of the instance, for example, the type of traffic it handles

Hovering the mouse over an instance displays a tooltip with detailed information, including the number of processes running in the instance, as well as the number of running threads for each process.

Use the button bar below the instances table to manage and configure the instances.

- **New**: Create a new instance. On a freshly installed Zorp, there are no instances — you have to create one first. See *Procedure 6.3.3, Creating a new instance (p. 107)* for details.
- **Delete**: Remove an instance.

> **Warning**
> Deleting an instance makes the services handled by the instance unaccessible.

- **Edit parameters**: Modify the name or parameters of the instance. See *Section 6.3.5, Instance parameters — general (p. 109)* for details. To modify the parameters of every instance, select ▾ > **Default parameters**.
- **Restart**: Restart the instance. To Reload, Restart, Stop, or Start an instance, click **Restart**▾ and select the desired action. These functions are needed after modifying the configuration of an instance. **Log level** sets the verbosity of logging. **Active connections** displays the connections currently handled by the instance (see *Section 6.8, Monitoring active connections (p. 186)* for details).

> **Tip**
> Use the Shift or the Control key to select and control multiple instances.

- **Arrow buttons**: Move the instance up or down in the list. When Zorp boots, the instances are started in the order they are listed.

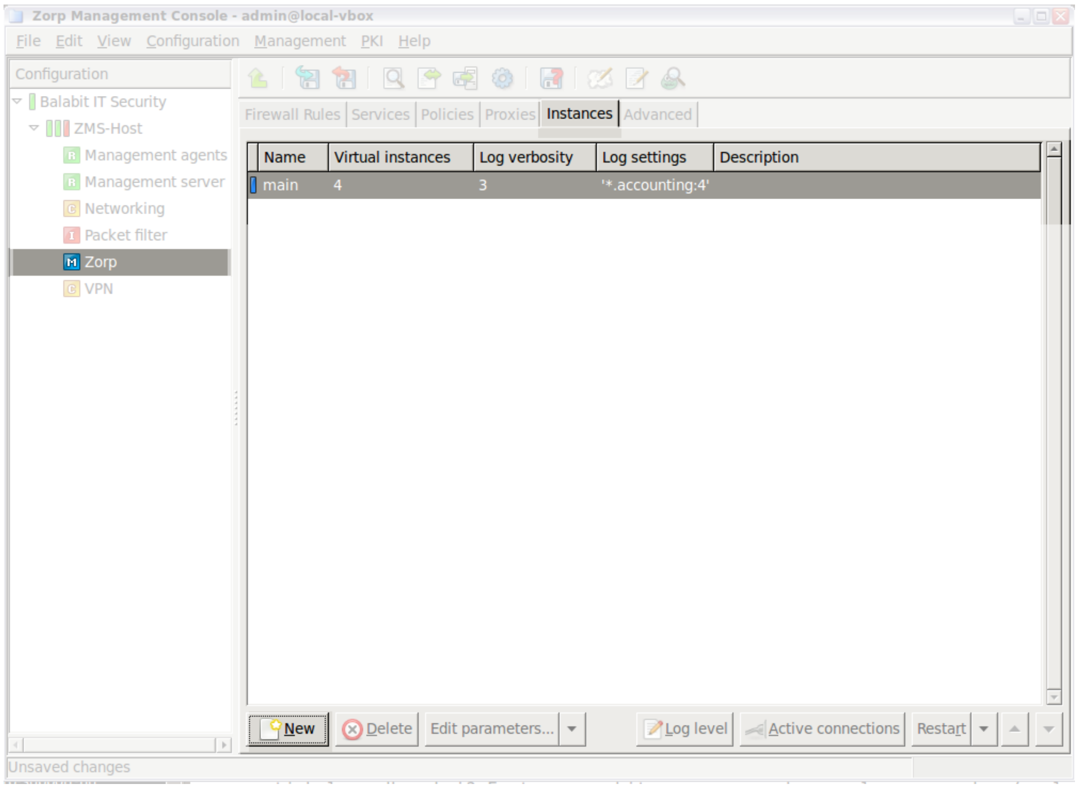
6.3.3. Procedure – Creating a new instance

To create a new instance on a Zorp firewall host, complete the following steps:

Step 1. Navigate to the **Instances** tab of the **Zorp** ZMC component on the Zorp host.

> **Note**
> If the **Zorp** and the **Packet Filter** components are not available on the selected host, you have to add them first. See *Procedure 3.2.1.3.1, Adding new configuration components to host (p. 22)* for details.

Step 2. Click the **New** button located below the **Instances** table.



Name	Virtual instances	Log verbosity	Log settings	Description
main	4	3	'*.accounting:4'	

Figure 6.17. Creating a new Zorp instance

Step 3. Enter a name for the new instance.

Note
Use informative names containing information about the direction and type of the traffic handled by the instance, for example, `intra_http` or `intra_pop3` referring to instances that handle HTTP and POP3 traffic coming from the intranet. Use direction names consistently, for example, include the source zone of the traffic.

Step 4. Describe the purpose of the instance in the **Description** field.

Step 5. To modify the parameters of the instance, uncheck the **Use default parameters** option and adjust the parameters as needed. For details on the available parameters, see *Section 6.3.5, Instance parameters — general (p. 109)*.

Step 6. Click **OK**.

6.3.4. Procedure – Configuring instances

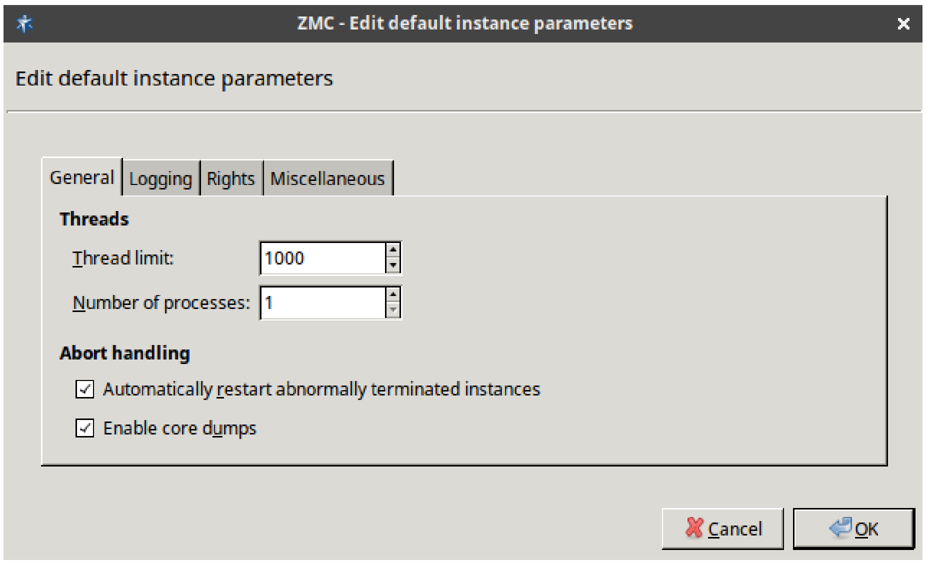
To modify the parameters of an instance, complete the following steps.

Step 1. Navigate to the **Instances** tab of the **Zorp** ZMC component on the Zorp host.

Step 2. Modify instances as described in either of the following options:

- To modify the configuration of every instance on the site, select ▾ > **Default parameters**.

- To modify the configuration of a single instance, select an instance and click **Edit parameters**. The **Edit Instance** window is displayed. Uncheck the **Use default parameters** option.


ZMC - Edit default instance parameters
Edit default instance parameters
General
Logging
Rights
Miscellaneous
Cancel
OK

Threads

Thread limit: 1000

Number of processes: 1

Abort handling

☑ Automatically restart abnormally terminated instances

☑ Enable core dumps

Figure 6.18. Edit instance parameters

Step 3. Adjust the settings as needed, then click **OK**. Instance parameters are grouped into four tabs. See *Section 6.3.5, Instance parameters — general (p. 109)* for details on the available parameters.

Step 4. Commit and upload the changes.

Step 5. To activate the changes, select the **Restart** button or the **Restart** > **Reload**.

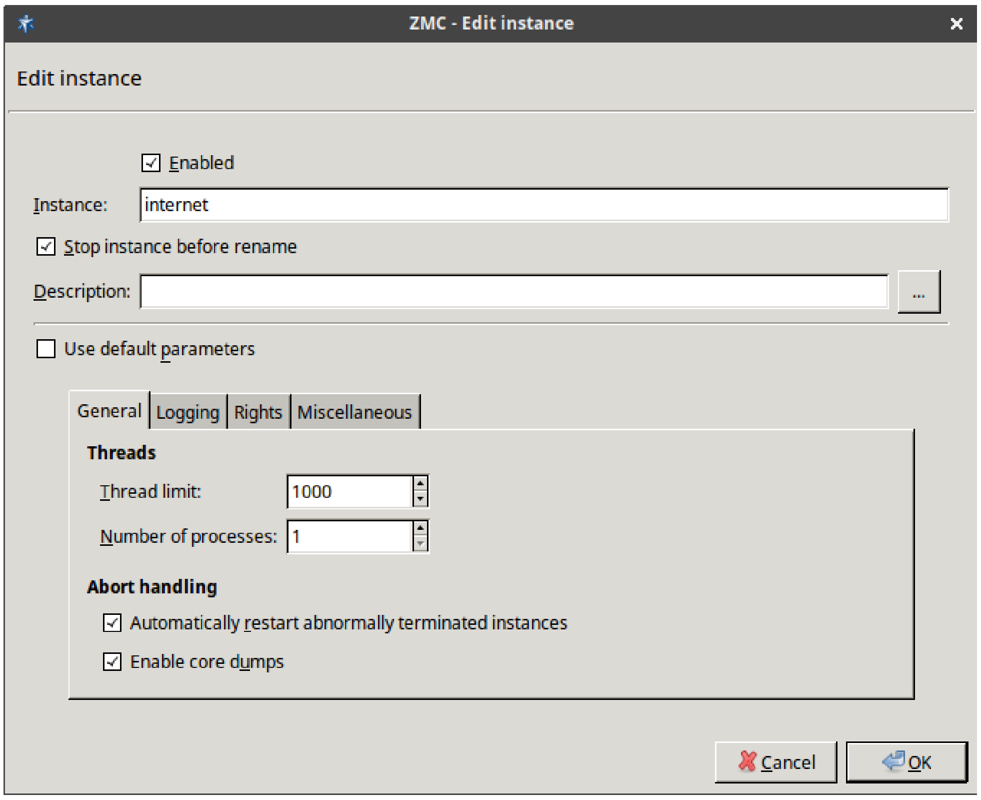
6.3.5. Instance parameters — general

To modify instance parameters, select **Zorp** > **Instance** > **Edit parameters**.

The following generic settings can be configured for those instance parameters that are *Active/Enabled*, consequently are visible in dark colour. *Inactive/Disabled* instance parameters are listed in light grey:

- **Instance**: It is the name of the instance.
- **Stop instance before rename**: When renaming an instance and this option is enabled, Zorp stops the instance, renames it, then starts the renamed instance.
- **Description**: The user can provide a description of the instance here.

The **General** tab has the following parameters:


ZMC - Edit instance

Edit instance

☑ Enabled

Instance: internet

☑ Stop instance before rename

Description:

☐ Use default parameters

General | Logging | Rights | Miscellaneous

Threads

Thread limit: 1000

Number of processes: 1

Abort handling

☑ Automatically restart abnormally terminated instances

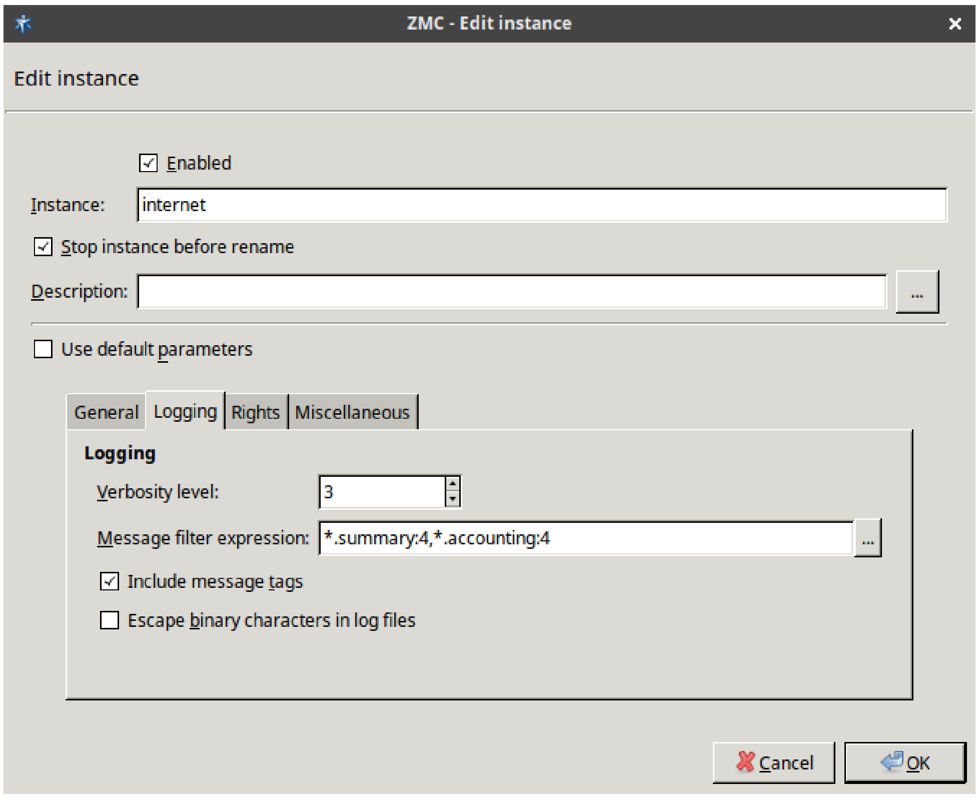
☑ Enable core dumps

Figure 6.19. General instance parameters

- **Thread limit**: It is the number of threads the instance can start. Set **Thread limit** according to the anticipated number of concurrent connections. Most of the active client requests require their own thread. If the **Thread limit** is too low, the clients will experience delays and refused connection attempts.
- **Number of processes**: It reflects the number of Zorp processes the instance can start. This setting determines the number of CPU cores that the instance can use. If our Zorp host has many CPUs, increase this value for instances that have high traffic. Note that the *`Thread limit`* and the *`Thread stack limit`* parameters are applied separately for each process. For details on increasing the number of running processes, see *Procedure 6.3.9, Increasing the number of running processes (p. 114)*.
  For every process, Zorp uses a certain amount of memory from the stack. At most, a process uses the default value of the stack size of the host (which is currently 8 MB for Ubuntu 18.04 LTS). Zorp uses this memory only when it is actually needed by the thread, it is not allocated in advance. Note that in Zorp version 5 and earlier, the stack size was only 1 MB, and you could increase it by using the **Thread stack limit** option.
- **Automatically restart abnormally terminated instances**: If enabled, Zorp automatically restarts instances that crash for some reasons.
- **Enable core dumps**: If enabled, Zorp automatically creates core dumps when a Zorp instance crashes for some reasons. Core dumps are special log files and are needed for troubleshooting.
  For more details on core dumps see *Section 10.11, Managing core dump files (p. 248)*.

6.3.6. Instance parameters — logging

Instance parameters can be set on the tabs of the **Edit Instance Parameters** window. The **Logging** tab has the following parameters:



ZMC - Edit instance

Edit instance

☑ Enabled

Instance: internet

☑ Stop instance before rename

Description:

☐ Use default parameters

Logging

Verbosity level: 3

Message filter expression: *.summary:4,*.accounting:4

☑ Include message tags

☐ Escape binary characters in log files

Figure 6.20. Instance parameters — logging

- **Verbosity level**: It is the general verbosity of the instance. It ranges from 0 to 9; the higher value means more detailed logging.

  Note
  Setting a high verbosity level (above 6) can dramatically decrease the performance. On level 9 Zorp logs the entire passing traffic.

  Tip
  The default verbosity level is 3, which logs every connection, error and violation without many details.

  Level 4 to 6 include protocol-specific information as well.

  Levels 7 to 9 are recommended only for troubleshooting and debugging purposes.

- **Message filter expression**: It sets the verbosity level on a per-category basis. Each log message has an assigned multi-level category, where levels are separated by a dot. For example, HTTP requests are logged under `http.request`. A log specification consists of a wildcard matching log category, a colon, and a number specifying the verbosity level of that given category. Separate log specifications with a comma. Categories match from left to right. For example, `http.*:5,core:3`. The last matching entry will be used as the verbosity of the given category. If no match is found the default verbosity is used.

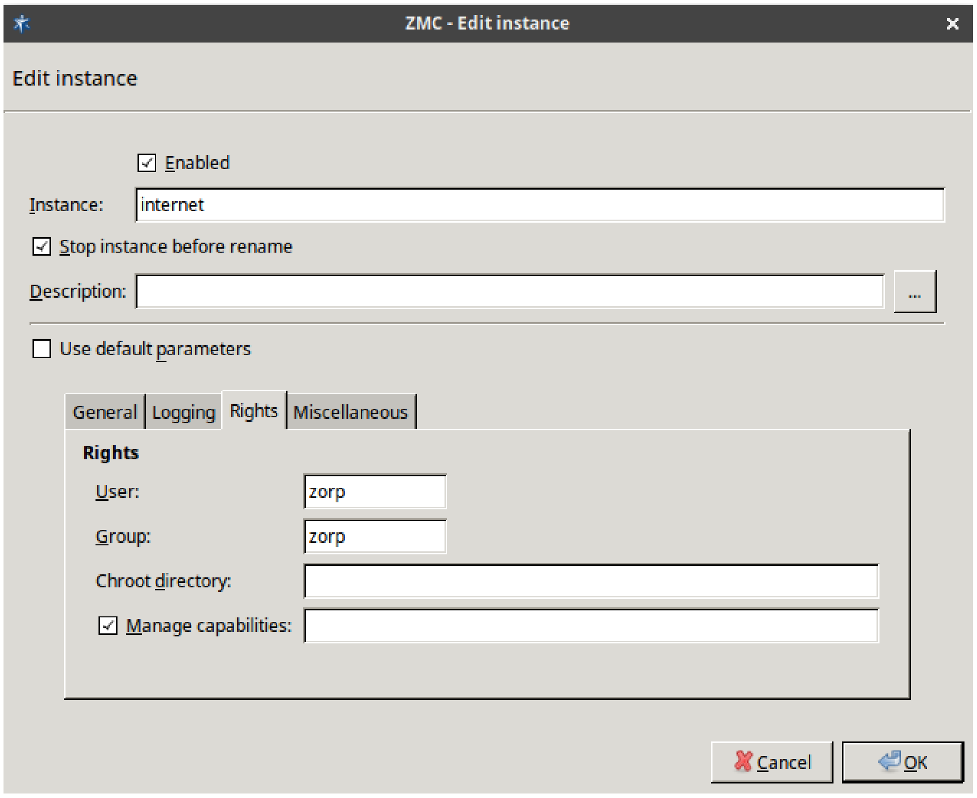
- **Include message tags**: It prepends a log category and a log level to each message.
- **Escape binary characters in log files**: It replaces non-printable characters with *`XX`* to avoid binary log files, also characters with codes lower than *`0x20`* or higher than *`0x7F`*.

Note
Customized logging can be very useful, but should be used with caution. Too many log specifications can decrease the overall performance of Zorp.

Example 6.4. Customized logging for HTTP accounting
The HTTP proxy logs accounting information into the `accounting` category. Requested URLs are logged on level 4. To log the URLs, but leave the general verbosity level on 3, add a new log specification to the **Message filter expression** list: *`http.accounting:4`*.

6.3.7. Instance parameters — Rights

Instance parameters can be set on the tabs of the **Edit Instance Parameters** window. The **Rights** tab has the following parameters:



☑ Enabled

Instance: internet

☑ Stop instance before rename

Description:

☐ Use default parameters

Rights

User:	zorp
Group:	zorp
Chroot directory:	
☑ Manage capabilities:	

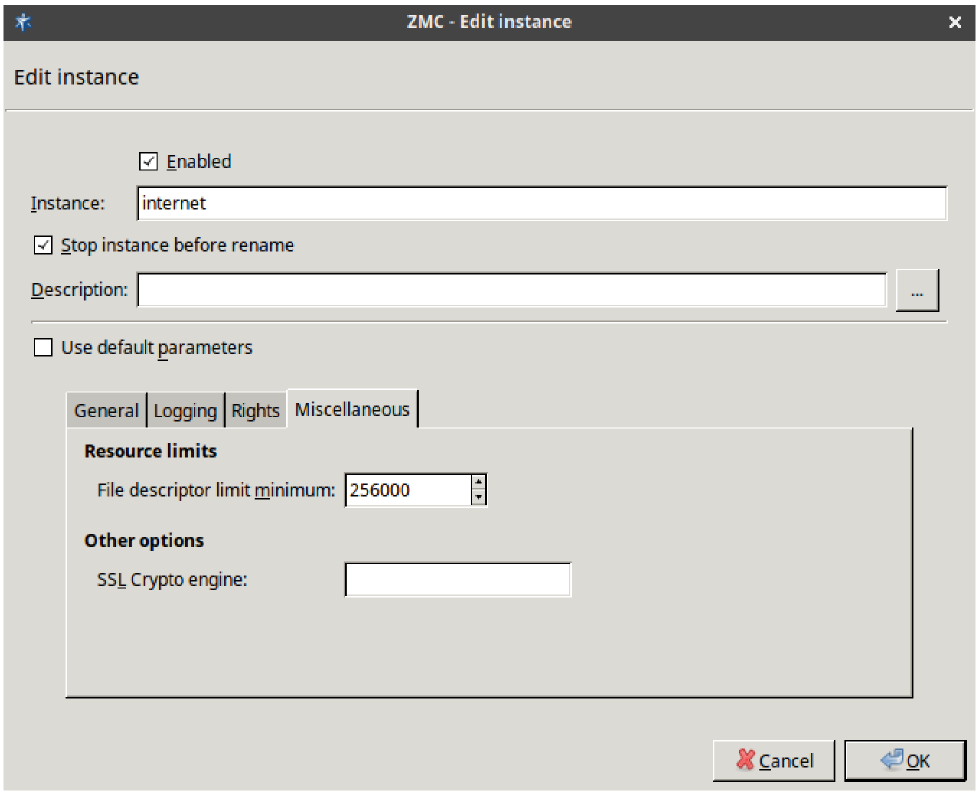
Figure 6.21. Instance parameters — Rights

Warning
Modify these settings only if you are completely aware of the necessary details, because bad configuration of rights can prevent Zorp from starting.

- **User**: Run Zorp as this user. By default, Zorp runs as the normal user `zorp`.
- **Group**: Run Zorp as a member of this group.
- **Chroot directory**: Change root to this directory before reading the configuration file.
- **Manage capabilities**: It is a whitespace-separated list of capability names to replace the ambient capability set of firewall processes.

6.3.8. Instance parameters — miscellaneous

Instance parameters can be set on the tabs of the **Edit Instance Parameters** window. The **Miscellaneous** tab has the following parameters:



☑ Enabled

Instance: internet

☑ Stop instance before rename

Description:

☐ Use default parameters

Resource limits

File descriptor limit minimum: 256000

Other options

SSL Crypto engine:

Figure 6.22. Instance parameters — miscellaneous

- **File descriptor limit minimum**: The number of open file descriptors that each firewall proccess could have.
- **SSL Crypto engine**: It defines the OpenSSL cryptographic engine to be used for hardware accelerated cryptographic support.

6.3.9. Procedure – Increasing the number of running processes

To increase the number of running processes in a Zorp instance, complete the following steps.

Steps:

Step 1. Navigate to the **Instances** tab of the **Zorp** ZMC component on the Zorp host.

Step 2. Select the instance you want to modify, and click **Edit parameters**.

Step 3. Uncheck the **Use default parameters** option.

Step 4. On the **General** tab, adjust the **Number of processes** option, and click **Ok**.

Step 5. Commit and upload your changes.

Step 6. **Reload** the instance.

Step 7. **Start** the instance.

6.4. Zorp services

Services define the traffic that can pass through the firewall. A service is not a software component, but a group of parameters that describe what kind of traffic should Zorp accept and how to handle the accepted traffic. The service specifies how thoroughly the traffic is analyzed (packet filter or application level), the protocol of the traffic (for example, HTTP, FTP, and so on), if the traffic is TLS-encrypted (and also related security settings like accepted certificates), NAT policies applied to the connections, and many other parameters.

Packet-filter services forward the incoming packets using the kzorp kernel module. Application-level services create two separate connections on the two sides of Zorp (client–Zorp, Zorp–server) different connections and analyze the traffic on the protocol level. Only application-level services can perform content filtering, authentication, and other advanced features.

Note
To allow IPSec traffic to pass Zorp, you must add packet filtering rules manually. See *Procedure 16.3.4, Forwarding IPSec traffic on the packet level (p. 441)* for details.

The following types of services are available in Zorp:

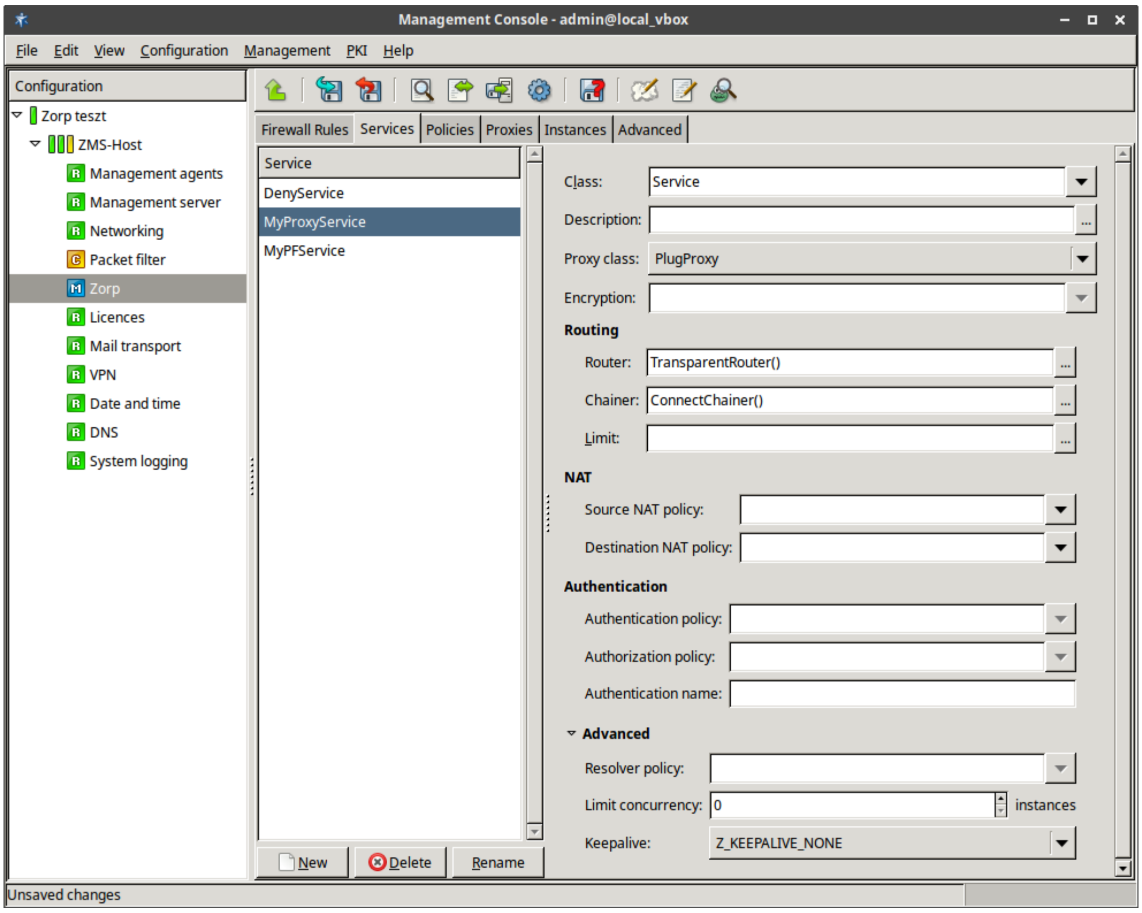
- **Service**: It inspects the traffic on application level using proxies. For the highest available security, use application-level inspection whenever possible. For details, see *Procedure 6.4.1, Creating a new service (p. 115)*
- **PFService**: It inspects the traffic only on packet level. Use packet-level filtering to transfer very large amount of UDP traffic (for example, streaming audio or video). For details, see *Procedure 6.4.2, Creating a new packet filtering Service (PFService) (p. 118)*.
- **DenyService**: It enables making a service unavailable for any reasons (for example, because accessing it is prohibited in certain zones), use **DenyService**. DenyService is a replacement for the umbrella zones of earlier Zorp versions. For details, see *Procedure 6.4.3, Creating a new DenyService (p. 119)*.
- **DetectorService**: It attempts to determine the protocol used in the connection from the traffic itself, and to start a specified service. Currently it can detect HTTP, SSH, and SSL traffic. For HTTPS connections, it can also select a service based on the certificate of the server. For details, see *Procedure 6.4.4, Creating a new DetectorService (p. 120)*.

Services are managed from the **Services** tab of the **Zorp** ZMC component. The left side of the tab displays the configured services, while the right side shows the parameters of the selected service. Use this tab to delete unwanted services, modify existing ones, or create new ones.

6.4.1. Procedure – Creating a new service

To create a new service that inspects a traffic on the application level, complete the following steps.

Step 1. Navigate to the **Services** tab of the **Zorp** ZMC component and click **New**.



Class: Service
Description:
Proxy class: PlugProxy
Encryption:
Routing
Router: TransparentRouter()
Chainer: ConnectChainer()
Limit:
NAT
Source NAT policy:
Destination NAT policy:
Authentication
Authentication policy:
Authorization policy:
Authentication name:
Advanced
Resolver policy:
Limit concurrency: 0 instances
Keepalive: Z_KEEPALIVE_NONE

Figure 6.23. Creating a new service

Step 2. Enter a name for the service into the opening dialog. Use clear, informative, and consistent service names. It is recommended to include the following information in the service name:

- source zones, indicating which clients may use the service (for example, *`intranet`*)
- the protocol permitted in the traffic (for example, *`HTTP`*)
- destination zones, indicating which servers may be accessed using the service (for example, *`Internet`*)

Tip
Name the service that allows internal users to browse the Web *`intra_HTTP_internet`*. Use dots to indicate child zones, for example, *`intra.marketing_HTTP_inter`*.

Step 3. Click ▾ in the **Class** field and select *`Service`*.

Step 4. In the **Proxy class** field, select the application-level proxy that will inspect the traffic. Only traffic corresponding to the selected protocol and the settings of the proxy class can pass the firewall.

Note
Zorp has many proxy classes available by default. These can be used as is, or can be customized if needed.

- For details on customizing proxy classes, see *Section 6.6, Proxy classes (p. 147)*.
- The settings and parameters of the proxy classes are detailed in the *Chapter 4, Proxies* in *Zorp Professional 7 Reference Guide*.
- To permit any type of Layer 7 traffic, select **PlugProxy**. The PlugProxy is a protocol-independent proxy.

Step 5. *Optional Step*: If the inspected traffic will be SSL- or TLS-encrypted, select the Encryption Policy to use in the **Encryption Policy** field. For details, see *Section 6.7.3, Encryption policies (p. 164)*.

Step 6. *Optional Step*: In the **Routing** section, select the method used to determine the IP address of the destination server. For details, see *Section 6.4.5, Routing — selecting routers and chainers (p. 122)*.

Step 7. *Optional Step*: In the **NAT** section, the Network Address Translation policy used to NAT the address of the client (SNAT), the server (DNAT), or both. For details, see *Section 6.7.5, NAT policies (p. 173)*.

Note
To remove a policy from the service, select the empty line from the combobox.

Note
NAT policies cannot be used in packet filtering services (PFServices) for IPv6 traffic.

Step 8. *Optional Step*: In the **Chainer** field, select the method used to connect to the destination server. See *Section 6.4.5, Routing — selecting routers and chainers (p. 122)* for details.

Step 9. *Optional Step*: To specify exactly which zones can be accessed using the service, click **Routing** > **Limit** > **...** and select the permitted zones. If this option is set, the target server must be located in the selected zones, otherwise Zorp will reject the connection.

Note
The zone set in the **Limit** option is the actual location of the target server. This is independent from the destination address of the client-side connection.

This option replaces the functionality of the `inband_services` parameter of the zone.

Step 10. *Optional Step*: In the **Authentication** section, select the authentication and authorization policies used to verify the identity of the client. See *Chapter 15, Connection authentication and authorization (p. 389)* for details.

Step 11. *Optional Step*: In the **Advanced** > **Resolver policy** field, select how Zorp should resolve the addresses of the client requests. See *Section 6.7.6, Resolver policies (p. 182)* for details.

Step 12. *Optional Step*: To limit how many clients can access the service at the same time, set the **Advanced** > **Limit concurrency** option. By default, Zorp does not limit the number of concurrent connections for a service (0).

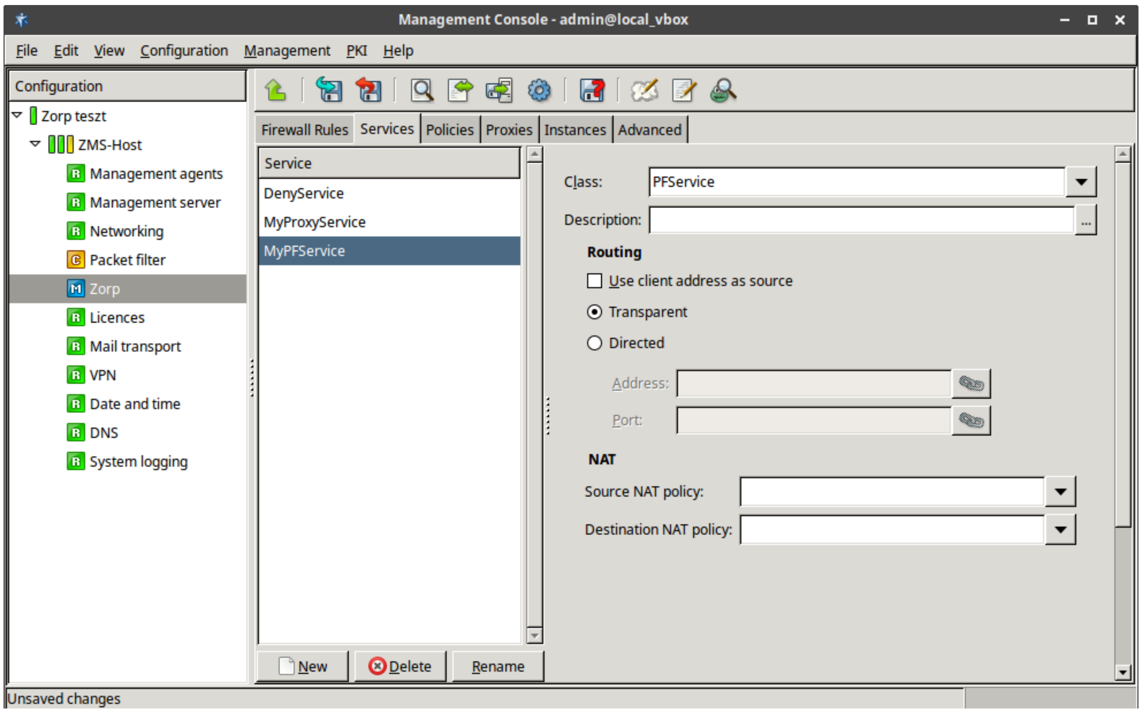
Step 13. *Optional Step*: To send keep-alive messages to the server, to the client, or to both, to keep the connection open even if there is no traffic, set the **Advanced** > **Keepalive** option to *`Z_KEEPALIVE_SERVER`*, *`Z_KEEPALIVE_CLIENT`*, or *`Z_KEEPALIVE_BOTH`*.

Step 14. Commit your changes.

6.4.2. Procedure – Creating a new packet filtering Service (PFService)

To create a new packet filter service that inspects a traffic on the packet level, complete the following steps.

Step 1. Navigate to the **Services** tab of the Zorp ZMC component and click **New**.



Class: PFService

Description:

Figure 6.24. Creating a new PFService

Step 2. Enter a name for the service into the opening dialog. Use clear, informative, and consistent service names. It is recommended to include the following information in the service name:

- source zones, indicating which clients may use the service (for example, *`intranet`*)
- the protocol permitted in the traffic (for example, *`HTTP`*)
- destination zones, indicating which servers may be accessed using the service (for example, *`Internet`*)

Tip
Name the service that allows internal users to browse the Web *`intra_HTTP_internet`*. Use dots to indicate child zones, for example, *`intra.marketing_HTTP_inter`*.

Step 3. Click ▾ in the **Class** field and select *`PFService`*.

Step 4. To spoof the IP address of the client in the server-side connection (so that the target server sees as if the connection originated from the client), select the **Use client address as source** option.

> **Note**
> For IPv6 traffic, the PFService will always spoof the client address, regardless of the setting of the **Use client address as source** option.

Step 5. To redirect the connection to a fixed address, select **Routing** > **Directed**, and enter the IP address and the port number of the target server into the respective fields. You can use links as well.

Step 6. *Optional Step*: In the **NAT** section, the Network Address Translation policy used to NAT the address of the client (SNAT), the server (DNAT), or both. For details, see *Section 6.7.5, NAT policies (p. 173)*.

> **Note**
> To remove a policy from the service, select the empty line from the combobox.

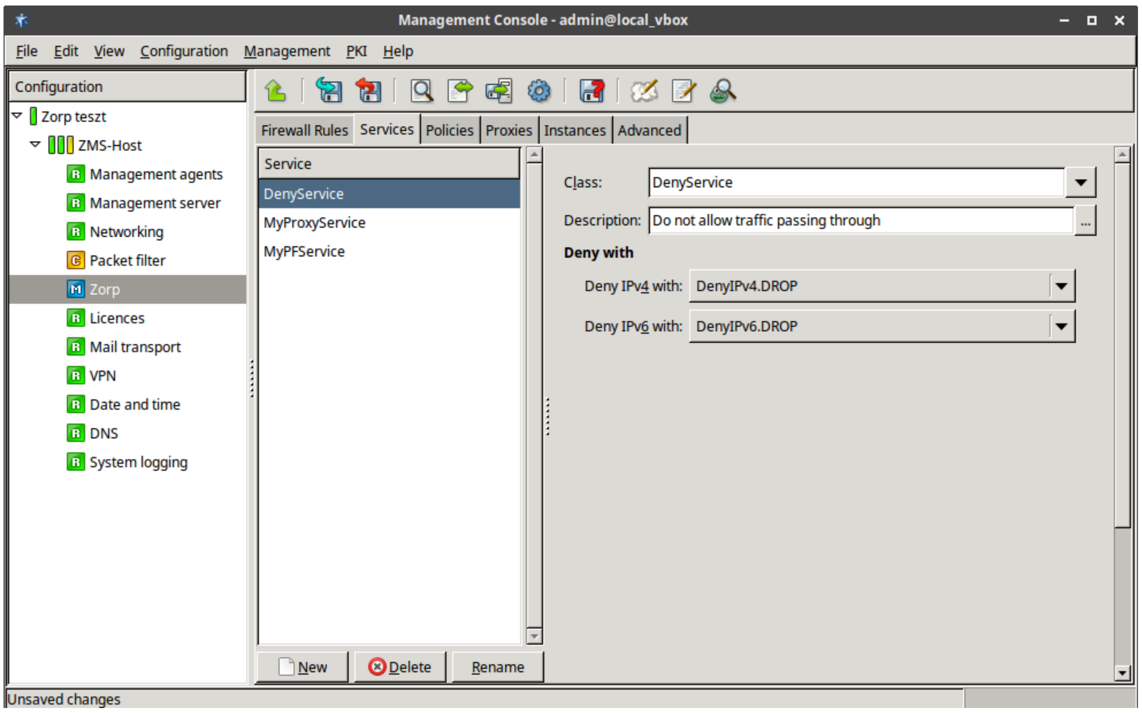
> **Note**
> NAT policies cannot be used in packet filtering services (PFServices) for IPv6 traffic.

Step 7. Commit your changes.

6.4.3. Procedure – Creating a new DenyService

To create a new DenyService that prohibits access to certain services, complete the following steps.

Step 1. Navigate to the **Services** tab of the Zorp ZMC component and click **New**.



Class: DenyService

Description: Do not allow traffic passing through

Deny with

Deny IPv4 with: DenyIPv4.DROP

Deny IPv6 with: DenyIPv6.DROP

Figure 6.25. Creating a new DenyService

Step 2. Enter a name for the service into the opening dialog. Use clear, informative, and consistent service names. It is recommended to include the following information in the service name:

- source zones, indicating which clients may use the service (for example, *`intranet`*)
- the protocol permitted in the traffic (for example, *`HTTP`*)
- destination zones, indicating which servers may be accessed using the service (for example, *`Internet`*)

Tip
Name the service that allows internal users to browse the Web *`intra_HTTP_internet`*. Use dots to indicate child zones, for example, *`intra.marketing_HTTP_inter`*.

Step 3. Click ▾ in the **Class** field and select *`DenyService`*.

Step 4. To specify how Zorp rejects the traffic matching a DenyService, use the **Deny IPv4 with** and **Deny IPv6 with** options. By default, Zorp simply drops the traffic without notifying the client.

Step 5. Commit your changes.

6.4.4. Procedure – Creating a new DetectorService

To create a new DetectorService that starts a service based on the traffic in the incoming connection, complete the following steps.

Step 1. Navigate to the **Services** tab of the Zorp ZMC component and click **New**.

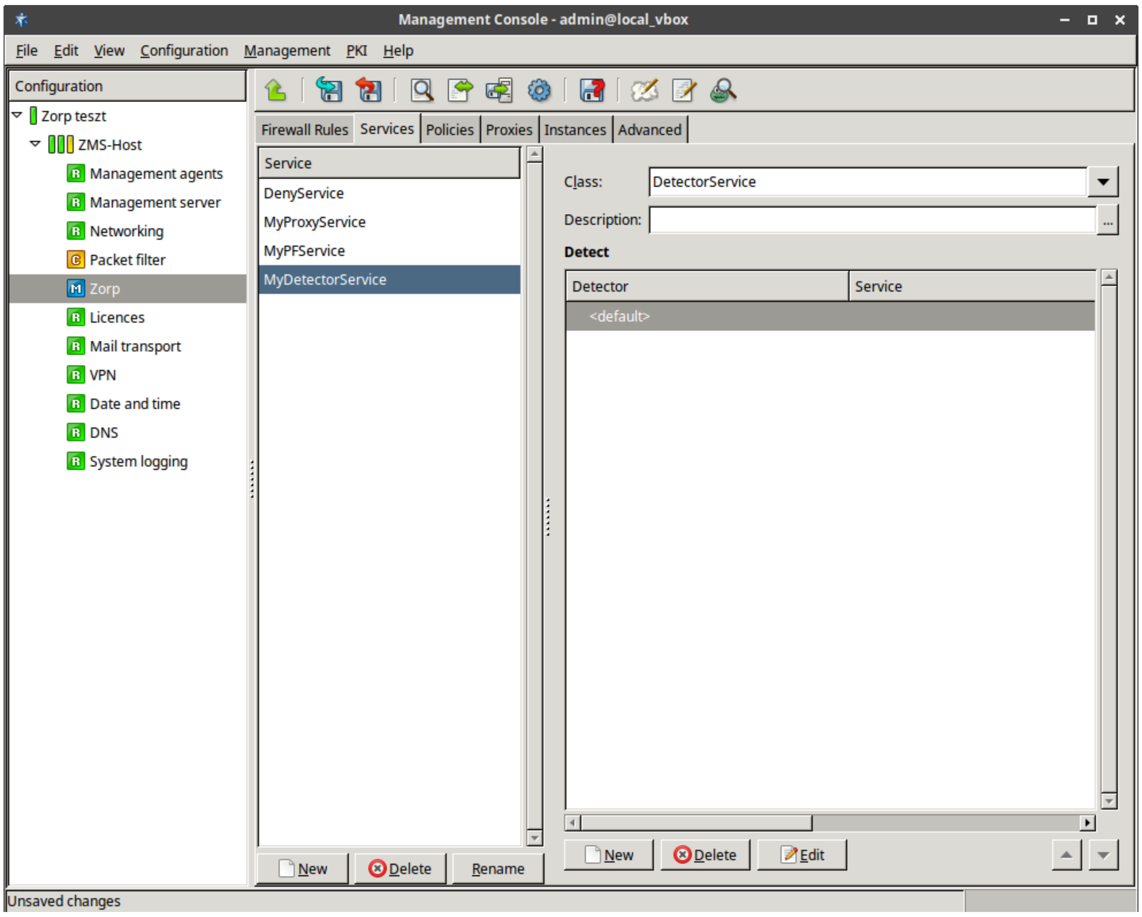


Figure 6.26. Creating a new DetectorService

Step 2. Enter a name for the service into the opening dialog. Use clear, informative, and consistent service names. It is recommended to include the following information in the service name:

- source zones, indicating which clients may use the service (for example, `intranet`)
- the protocol permitted in the traffic (for example, *HTTP*)
- destination zones, indicating which servers may be accessed using the service (for example, *Internet*)

Tip
Name the service that allows internal users to browse the Web *intra_HTTP_internet*. Use dots to indicate child zones, for example, *intra.marketing_HTTP_inter*.

Step 3. In the **Routing** section, select the TransparentRouter option.

Step 4. Click ▾ in the **Class** field and select *DetectorService*.

Step 5. Commit your changes.

Step 6. Navigate to **Zorp** > **Firewall Rules**, and create a firewall rule that uses the DetectorService you created in the previous steps.

Step 7. Click **New**, select a DetectorPolicy, and select a service that Zorp will start if the traffic matches the DetectorPolicy. If you add more DetectorPolicy-Service pairs, Zorp will evaluate them in order, and start the service set for the first matching DetectorPolicy. If none of the DetectorPolicies match the traffic, Zorp terminates the connection.

Note
When using a DetectorService, establishing the connection is slower, because Zorp needs to evaluate the content of the traffic before starting the appropriate service. If the rate of incoming connection requests that use the DetectorService is high, the clients may experience performance problems during connection startup. Note that using a DetectorService has no effect on the performance after the connection has been established.

6.4.5. Routing — selecting routers and chainers

Routers define the target IP address and the port for the traffic. The default router, called TransparentRouter, uses the IP address requested by the client. The destination selected by a router may be later overridden by the proxy (if the `Target address overridable by the proxy` option of the router is enabled) and the chainer.

Routers suggest the destination IP; chainers establish the connection with the selected destination. The default ConnectChainer simply connects the destination selected by the router.

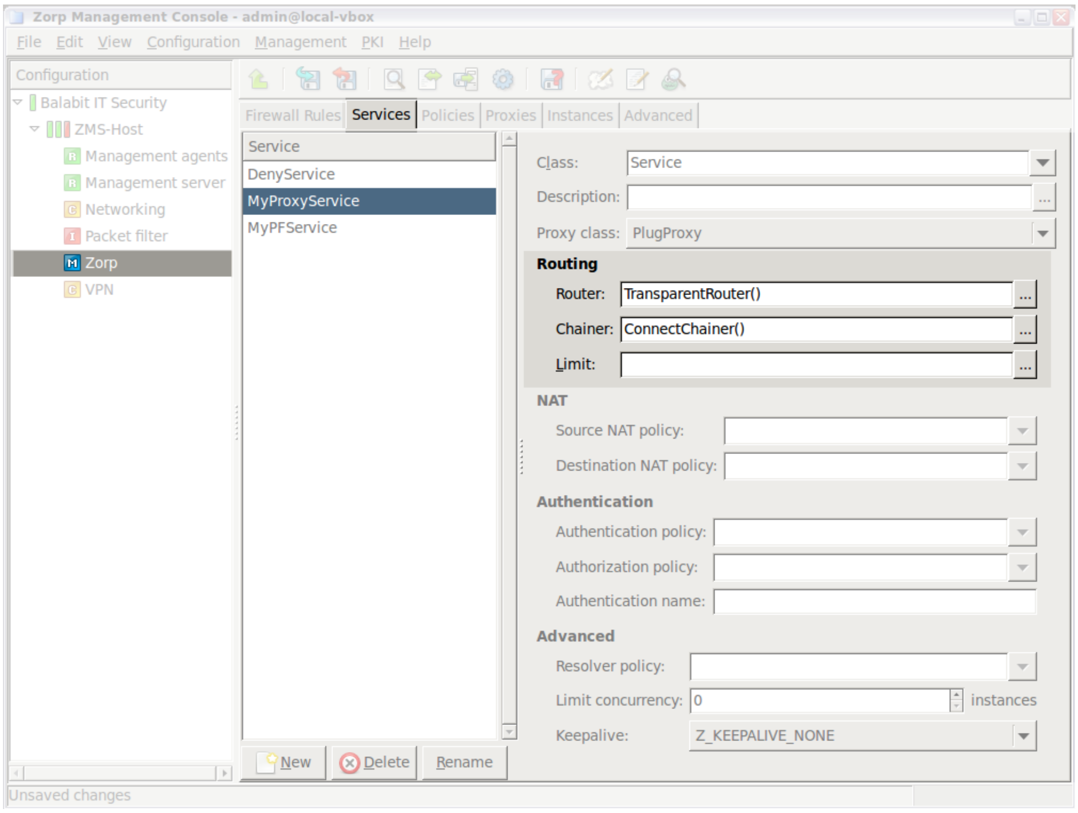
6.4.5.1. Procedure – Setting routers and chainers for a service

To set a router or a chainer for a service, complete the following steps.

Step 1. Navigate to the **Services** tab of the **Zorp** ZMC component and select the service to modify.

Step 2. Click **Router** > **...** to select and configure a router. By default, the TransparentRouter is selected. The following routers are available:

- *TransparentRouter*
- *DirectedRouter*
- *InbandRouter*



Class: Service
Description:
Proxy class: PlugProxy

Routing
Router: TransparentRouter()
Chainer: ConnectChainer()
Limit:

NAT
Source NAT policy:
Destination NAT policy:

Authentication
Authentication policy:
Authorization policy:
Authentication name:

Advanced
Resolver policy:
Limit concurrency: 0 instances
Keepalive: Z_KEEPALIVE_NONE

Figure 6.27. Configuring routers and chainers

Step 3. Click **Chainer** > **...** to select and configure a chainer. By default, the ConnectChainer is selected. The following chainers are available:

- *ConnectChainer*
- *FailoverChainer*
- *RoundRobinChainer*
- *SidestackChainer*
- *AvailabilityChainer*
- *RoundRobinAvailabilityChainer*

Step 4. Commit your changes.

6.4.5.2. TransparentRouter

TransparentRouter does not modify the IP address and the TCP/UDP port number requested by the client; the same parameters are used on the server side.

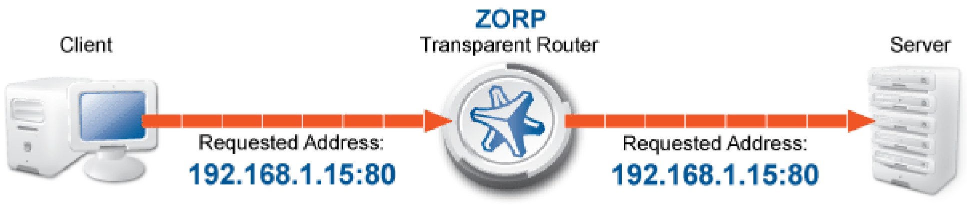
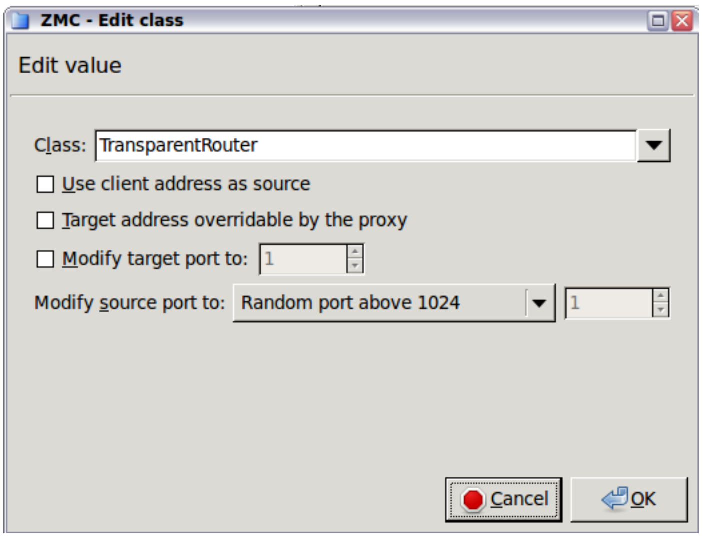


Figure 6.28. Using the TransparentRouter


ZMC - Edit class

Edit value

Class: TransparentRouter

☐ Use client address as source

☐ Target address overridable by the proxy

☐ Modify target port to: 1

Modify source port to: Random port above 1024 1

Figure 6.29. Configuring TransparentRouter

The following parameters can be configured for TransparentRouter:

Use client address as source

By default, Zorp uses its own IP address in the server-side connections: the server does not see the IP address of the original client. By selecting this option, Zorp mimics the original address of the client. Use this option if the server uses IP-based authentication, or the address of the client must appear in the server logs.

Note
This option was called **Forge address** in earlier versions of Zorp.

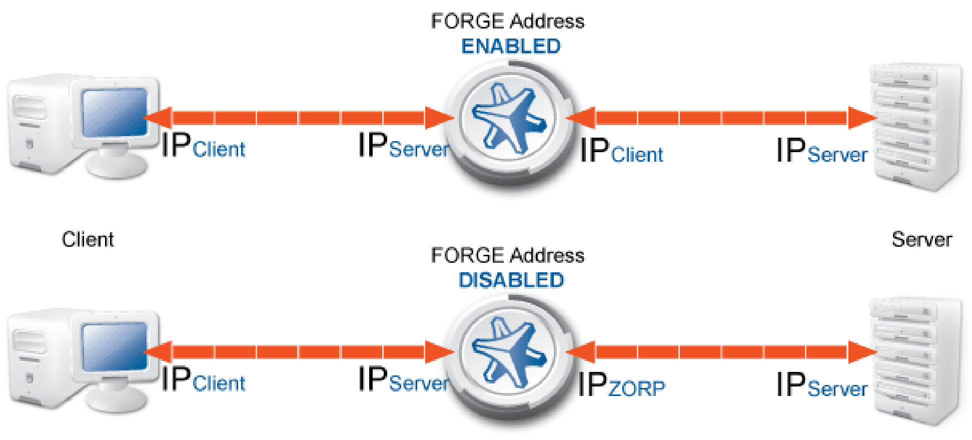


Figure 6.30. Using the client address in server-side connections

Note
The IP address of the client is related to the source NAT (SNAT) policy used for the service: using SNAT automatically enables the **Use client address as source** option in the router.

Target address overridable by the proxy

If this option is selected and the data stream in the connection contains routing information, than the address specified in the data stream is used as the destination address of the server-side connection.

Example 6.5. Overriding the target port SQLNetProxy
The Oracle SQLNet protocol can request port redirection within the protocol. Configure a service using the SQLNetProxy and the **Target address overridable by the proxy** router option. When a client first connects to the Oracle server, the connection is established to the IP address and the port selected by the router. However, the server can send a redirect request to the client, and the router has to reconnect to the port specified in the request of the Oracle server. This procedure is performed transparently to the client.

Note
The **Target address overridable by the proxy** option cannot be used with InbandRouter.

This option was called **Overridable** in earlier versions of Zorp.

Modify target port

Use the **Modify target port** option to connect to a different port of the server.

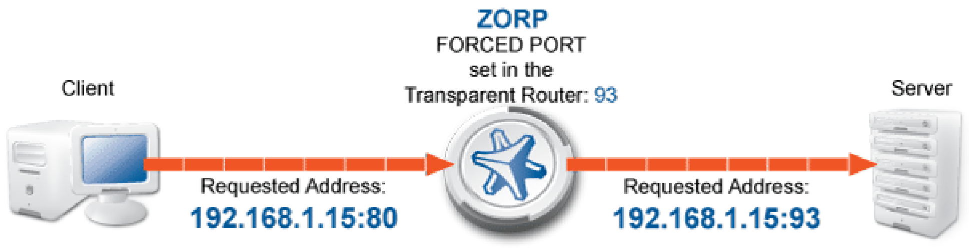


Figure 6.31. Using TransparentRouter with the Modify target port option

Modify source port
This option defines the source port that Zorp uses in the server-side connection. The following options are available:

- *Random port above 1024*: Select a random port between 1024 and 65535. This is the default behavior of every router.
- *Random port in the same group*: Select a random port in the same group as the port used by the client. The following groups are defined: 0-513, 514-1024, 1025–.
- *Client port*: Use the same port as the client.
- *Specified port*: Use the port set in the spinbutton.

6.4.5.3. DirectedRouter

DirectedRouter directs all connections to fixed addresses, regardless of the address requested by the client.

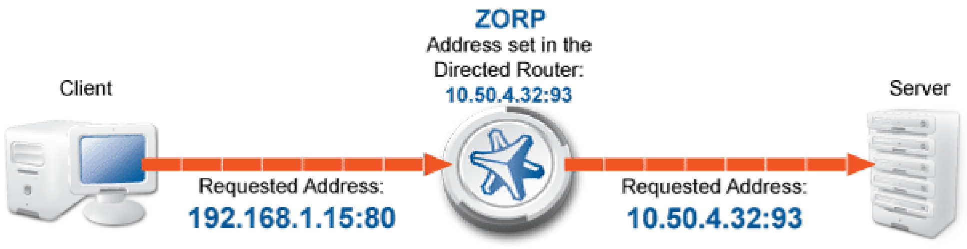
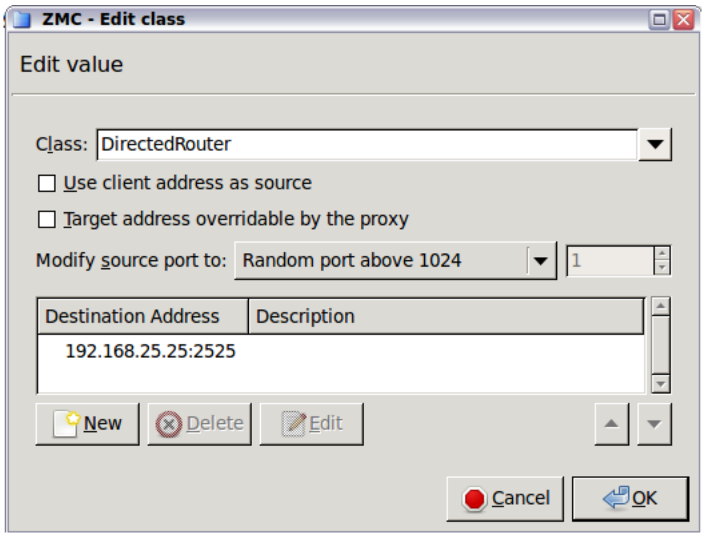


Figure 6.32. Using the DirectedRouter


ZMC - Edit class

Edit value

Class: DirectedRouter

☐ Use client address as source

☐ Target address overridable by the proxy

Modify source port to: Random port above 1024 | 1

Destination Address	Description
192.168.25.25:2525	

Figure 6.33. Configuring DirectedRouter

To add a destination address, click **New**, and enter the IP address and the port number of the server to connect to. If you set additional addresses, and the first address is unreachable, Zorp tries to connect to the next server. It is also possible to connect to the listed destinations in a round-robin fashion, see *Section 6.4.5.7, RoundRobinChainer (p. 131)* for details.

Tip
Use DirectedRouter for servers publicly available in the DMZ. That way outsiders do not know the real IP addresses of the servers — the servers are not even required to have public, routable IP addresses.

DirectedRouter has the following options:

Use client address as source

By default, Zorp uses its own IP address in the server-side connections: the server does not see the IP address of the original client. By selecting this option, Zorp mimics the original address of the client. Use this option if the server uses IP-based authentication, or the address of the client must appear in the server logs.

Note
This option was called **Forge address** in earlier versions of Zorp.

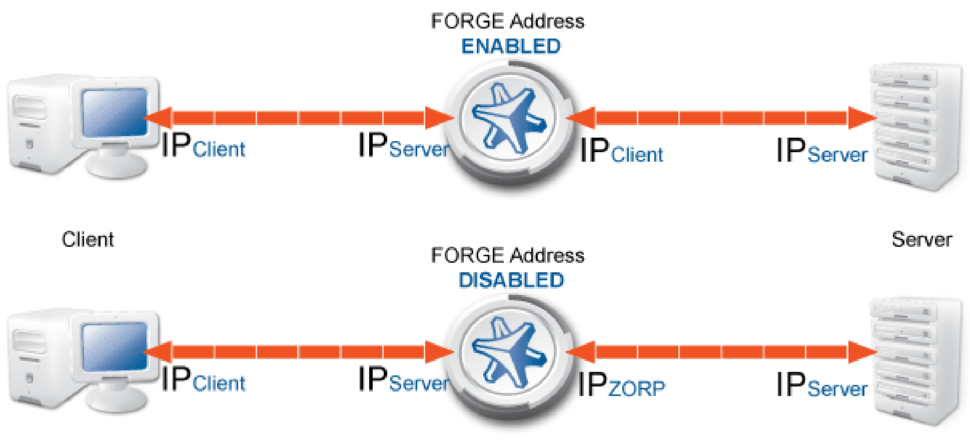


Figure 6.34. Using the client address in server-side connections

Note
The IP address of the client is related to the source NAT (SNAT) policy used for the service: using SNAT automatically enables the **Use client address as source** option in the router.

Target address overridable by the proxy

If this option is selected and the data stream in the connection contains routing information, than the address specified in the data stream is used as the destination address of the server-side connection.

Example 6.6. Overriding the target port SQLNetProxy
The Oracle SQLNet protocol can request port redirection within the protocol. Configure a service using the SQLNetProxy and the **Target address overridable by the proxy** router option. When a client first connects to the Oracle server, the connection is established to the IP address and the port selected by the router. However, the server can send a redirect request to the client, and the router has to reconnect to the port specified in the request of the Oracle server. This procedure is performed transparently to the client.

Note
The **Target address overridable by the proxy** option cannot be used with InbandRouter.

This option was called **Overridable** in earlier versions of Zorp.

Modify source port

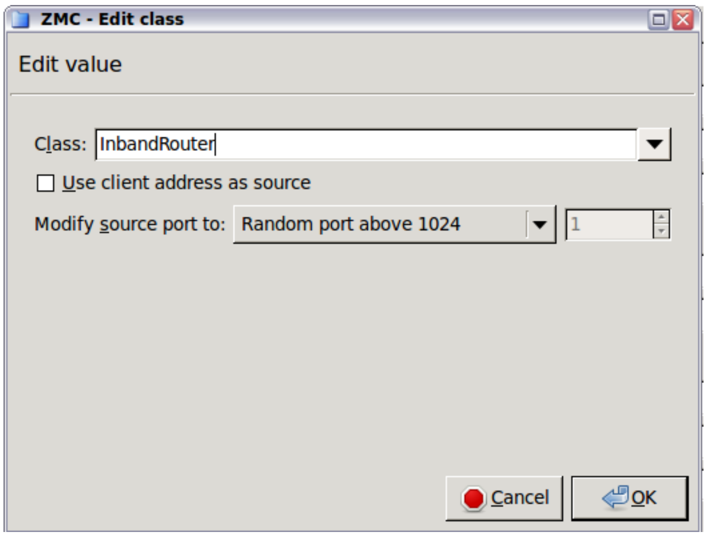
This option defines the source port that Zorp uses in the server-side connection. The following options are available:

- *Random port above 1024*: Select a random port between 1024 and 65535. This is the default behavior of every router.
- *Random port in the same group*: Select a random port in the same group as the port used by the client. The following groups are defined: 0-513, 514-1024, 1025–.
- *Client port*: Use the same port as the client.
- *Specified port*: Use the port set in the spinbutton.

Note
This option was called **Forge port** in earlier versions of Zorp.

6.4.5.4. InbandRouter

The InbandRouter determines the target address from the information embedded in the transferred protocol. This is possible only for protocols that can have routing information within the data stream. Zorp can use InbandRouter with the HTTP and FTP protocols.



Class: InbandRouter
☐ Use client address as source
Modify source port to: Random port above 1024 | 1

Figure 6.35. Configuring InbandRouter

The InbandRouter has the following options:

Use client address as source

By default, Zorp uses its own IP address in the server-side connections: the server does not see the IP address of the original client. By selecting this option, Zorp mimics the original address of the client. Use this option if the server uses IP-based authentication, or the address of the client must appear in the server logs.

Note
This option was called **Forge address** in earlier versions of Zorp.

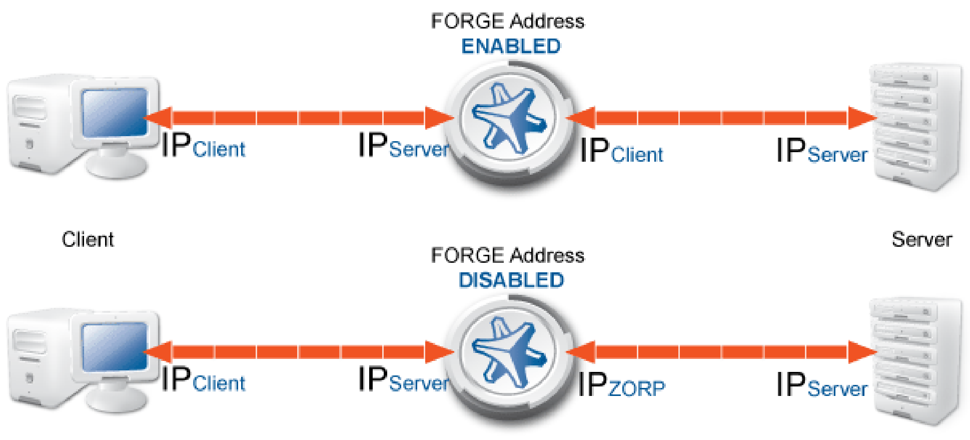


Figure 6.36. Using the client address in server-side connections

Note
The IP address of the client is related to the source NAT (SNAT) policy used for the service: using SNAT automatically enables the **Use client address as source** option in the router.

Modify source port

This option defines the source port that Zorp uses in the server-side connection. The following options are available:

- *Random port above 1024*: Select a random port between 1024 and 65535. This is the default behavior of every router.
- *Random port in the same group*: Select a random port in the same group as the port used by the client. The following groups are defined: 0-513, 514-1024, 1025–.
- *Client port*: Use the same port as the client.
- *Specified port*: Use the port set in the spinbutton.

Note
This option was called **Forge port** in earlier versions of Zorp.

6.4.5.5. ConnectChainer

ConnectChainer attempts to connect to the destination defined by the router. It terminates the connection if the destination server is unreachable. ConnectChainer has the following options:

Connection timeout

Assume that the server is unreachable if it cannot be connected for `Connection timeout` seconds.

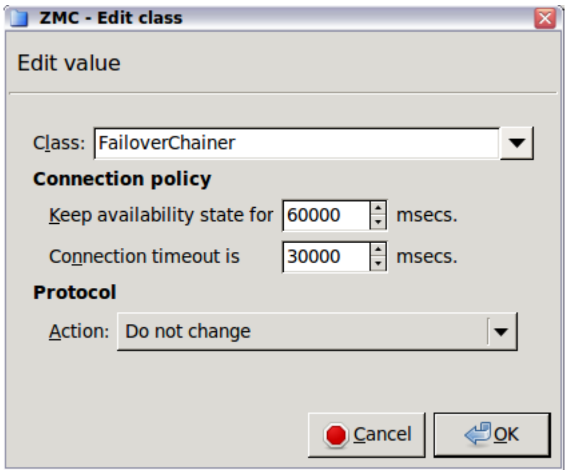
Protocol action

It defines the type of the protocol used in the server-side connection. The default is to use the same protocol as on the client side, but Zorp can enforce the use of TCP or UDP protocol.

6.4.5.6. FailoverChainer

FailoverChainer is similar to *ConnectChainer*, and attempts to connect to the destination defined by the router. However, if the destination is unreachable and the service uses *DirectedRouter*, Zorp attempts to connect to the next destination set in the router.

Tip
FailoverChainer can be used to implement a simple high availability support for the protected servers.



Class: FailoverChainer

Connection policy

Keep availability state for 60000 msecs.

Connection timeout is 30000 msecs.

Protocol

Action: Do not change

Figure 6.37. Configuring FailOverChainer

FailoverChainer has the following options:

Keep availability state for
Zorp does not try to connect to an unreachable server until the time set in the **Keep availability state for** option expires.

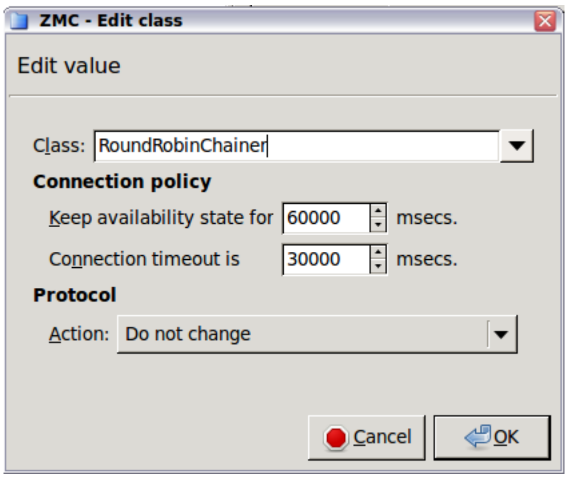
Connection timeout
Assume that the server is unreachable if it cannot be connected for `Connection timeout` seconds.

Protocol action
It defines the type of the protocol used in the server-side connection. The default is to use the same protocol as on the client side, but Zorp can enforce the use of TCP or UDP protocol.

6.4.5.7. RoundRobinChainer

RoundRobinChainer is similar to *FailoverChainer*, but Zorp directs each incoming connection to the next available server. That way the load is balanced between the servers set in the destination list of the router. Use the RoundRobinChainer together with the *DirectedRouter*.

Tip
RoundRobinChainer can be used to implement simple load balancing for the protected servers.


ZMC - Edit class

Edit value

Class: RoundRobinChainer

Connection policy

Keep availability state for 60000 msecs.

Connection timeout is 30000 msecs.

Protocol

Action: Do not change

Figure 6.38. Configuring RoundRobinChainer

RoundRobinChainer has the following options:

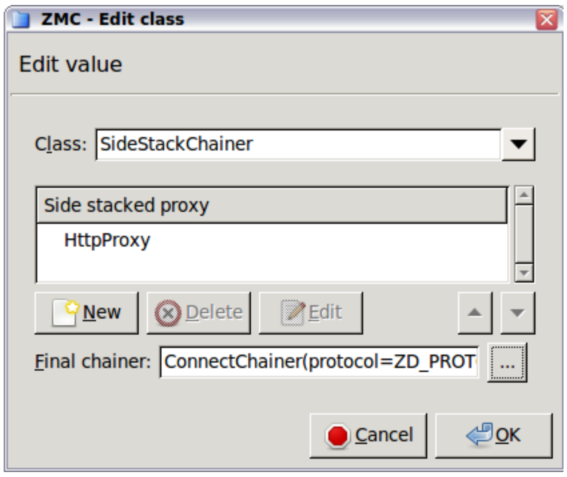
Keep availability state for
Zorp does not try to connect to an unreachable server until the time set in the **Keep availability state for** option expires.

Connection timeout
Assume that the server is unreachable if it cannot be connected for `Connection timeout` seconds.

Protocol action
It defines the type of the protocol used in the server-side connection. The default is to use the same protocol as on the client side, but Zorp can enforce the use of TCP or UDP protocol.

6.4.5.8. SidestackChainer

SidestackChainer does not connect to the server, but passes the traffic to the specified proxy. It is possible to sidestack several proxies that way. Zorp connects to the server using a regular chainer after the proxy processes the traffic. SidestackChainer has the following options:



Class: SideStackChainer

Side stacked proxy
HttpProxy

Final chainer: ConnectChainer(protocol=ZD_PROT

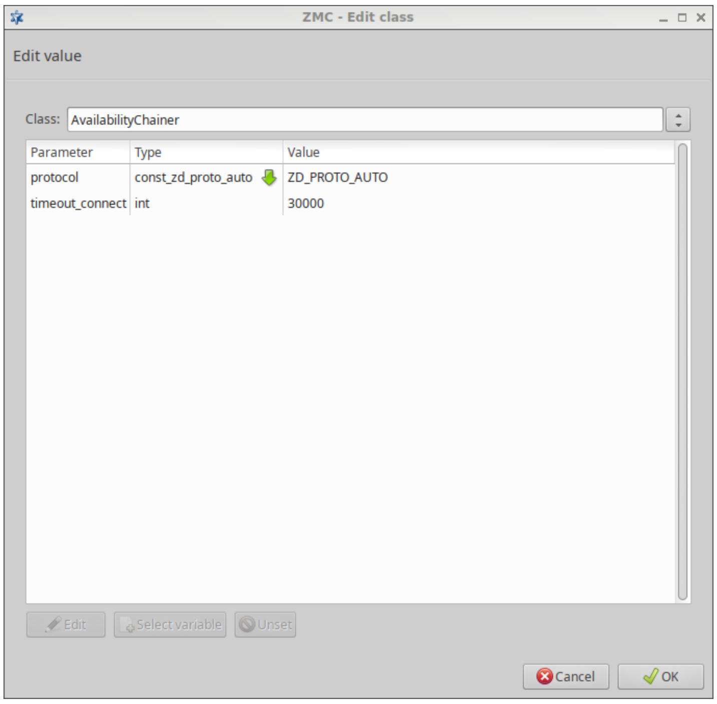
Figure 6.39. Configuring SidestackChainer

Side-stacked proxy
It is the proxy class that will process the traffic. To select a proxy, click **New**, select a proxy class, then click **Select**.

Final chainer
This is the chainer used when connecting the server.

6.4.5.9. AvailabilityChainer

The AvailabilityChainer enables establishing connection with multiple target addresses and using information from the Availability Checker daemon. The Availability Checker daemon monitors the servers if they are up or down, and based on that information the AvailabilityChainer can automatically target adresses which are in up state. The AvailabilityChainer connects to target hosts in the order they have been specified. Use the AvailabilityChainer together with the *DirectedRouter*.



Class: AvailabilityChainer

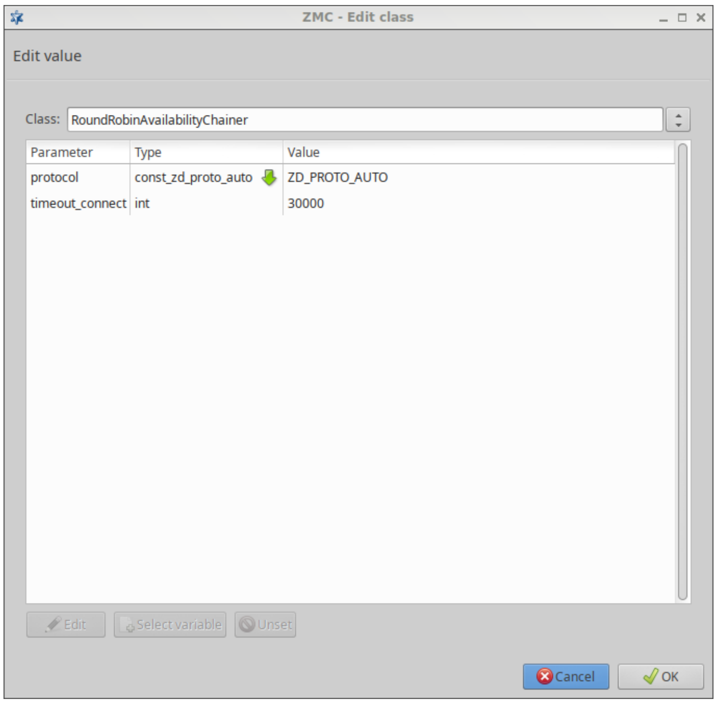
Parameter	Type	Value
protocol	const_zd_proto_auto	ZD_PROTO_AUTO
timeout_connect	int	30000

Figure 6.40. Configuring AvailabilityChainer

To enable the AvailabilityChainer, the Availability Checker component has to be activated and configured. For more information on the Availability Checker component, see *Section 12.7, Availability Checker (p. 319)*.

6.4.5.10. RoundRobinAvailabilityChainer

The RoundRobinAvailabilityChainer establishes a load balancing solution, using information from the Availability Checker daemon. The Availability Checker daemon monitors the servers if they are up or down, and based on that information the RoundRobinAvailabilityChainer can automatically target the next available host in up state. The RoundRobinChainer connects to hosts in an order based on the information from the Availability Checker daemon. Use the RoundRobinAvailabilityChainer together with the *DirectedRouter*.



Class: RoundRobinAvailabilityChainer

Parameter	Type	Value
protocol	const_zd_proto_auto	ZD_PROTO_AUTO
timeout_connect	int	30000

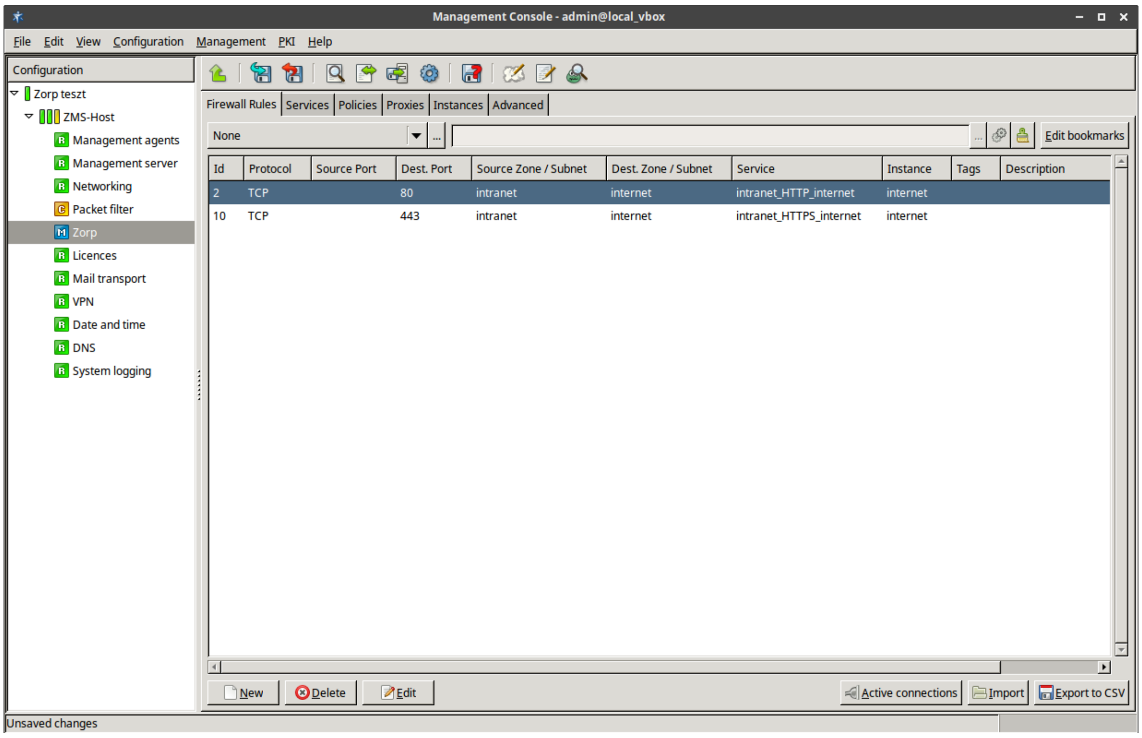
Figure 6.41. Configuring RoundRobinAvailabilityChainer

To enable the RoundRobinAvailabilityChainer, the Availability Checker component has to be activated and configured. For more information on the Availability Checker component, see *Section 12.7, Availability Checker (p. 319)*.

6.5. Configuring firewall rules

6.5.1. Understanding Zorp firewall rules

Zorp firewall rules are managed on the **<Host> > Zorp > Firewall Rules** page. The following information is displayed for every rule:



Id	Protocol	Source Port	Dest. Port	Source Zone / Subnet	Dest. Zone / Subnet	Service	Instance	Tags	Description
2	TCP		80	intranet	internet	intranet_HTTP_internet	internet		
10	TCP		443	intranet	internet	intranet_HTTPS_internet	internet		

Figure 6.42. Configuring Firewall rules

Note
Not every column is displayed by default. To show or hide a particular column, right-click on the header of the table and select the column from the menu.

Whether a certain rule is active or not is visible by its colour, that is, it is dark-grey if the rule is active and it is light-grey if the rule is inactive.

- **ID**: It is the unique ID number of the firewall rule.
- **Tags**: These are the tags (labels) assigned to the firewall rule. For details on assigning tags to rules, see *Procedure 6.5.5, Tagging firewall rules (p. 144)*.
- **Protocol**: It is the transport protocol used in the connection. This is the protocol used in the transport layer (Layer 4) of the OSI model. For example, TCP, UDP, ICMP, and so on.
- **VPN**: The rule permits traffic only from the listed VPN connections (or IPSec connections with the specified Request ID).
- **Source Zone/Subnet**: The rule permits traffic only for the clients of the listed zones and subnets.
- **Source Port**: The rule permits traffic only for connections targeting the listed ports of the firewall host.
- **Destination Zone/Subnet**: The rule permits traffic only for connections that target addresses of the listed zones and subnets.

- **Destination Interface/Group**: The rule permits traffic only for connections that target an existing IP address of the selected interface (or interface group) of the firewall host. This parameter can be used to provide nontransparent service on an interface that received its IP address dynamically.
- **Destination Port**: The rule permits traffic only for connections that target the listed ports of the destination address.
- **Service**: The name of the service is provided here used to inspect the traffic.
- **Instance**: The service started by the rule belongs to the instance shown.
- **Description**: It provides a description of the rule.
- **ICMP type and code**: ICMP type determines what the ICMP packet is used for. If the type does not have any codes defined, the code field is set to zero.

6.5.1.1. Evaluating firewall rules

When Zorp receives a connection request from a client, it tries to select a rule matching the parameters of the connection. The following parameters are considered.

Name in ZMC	Name in policy.py
VPN	*reqid*
Source Interface	*src_iface*
Source Interface Group	*src_ifgroup*
Protocol	*proto*
Source Port	*src_port*
Destination Port	*dst_port*
Source Subnet	*src_subnet*
Source Zone	*src_zone*
Destination Subnet	*dst_subnet*
Destination Interface	*dst_iface*
Destination Interface Group	*dst_ifgroup*
Destination Zone	*dst_zone*

Table 6.2. Evaluated Rule parameters

Zorp selects the rule that most specifically matches the connection. Selecting the most specific rule is based on the following method.

- The order of the rules is not important.
- The parameters of the connection act as filters: if you do not set any parameters, the rule will match any connection.
- If multiple connections would match a connection, the rule with the most-specific match is selected.

For example, you have configured two rules: the first has the *`Source Zone`* parameter set as the *`office`* (which is a zone covering all of your client IP addresses), the second has the *`Source Subnet`* parameter set as *`192.168.15.15/32`*. The other parameters of the rules are the same. If a connection request arrives from the *`192.168.15.15/32`* address, Zorp will select the second rule. The first rule will match every other client request.

- Zorp considers the parameters of a connection in groups. The first group is the least-specific, the last one is the most-specific. The parameter groups are listed below.
- The parameter groups are linked with a logical AND operator: if parameters of multiple groups are set in a rule, the connection request must match a parameter of every group. For example, if both the *`Source Interface`* and *`Destination Port`* are set, the connection must match both parameters.
- Parameters within the same group are linked with a logical OR operator: if multiple parameters of a group are set for a rule, the connection must match any one of the parameters. If there are multiple similar rules, the rule with the most specific parameter match for the connection will be selected.

Note
In general, avoid using multiple parameters of the same group in one rule, as it may lead to undesired side-effects. Use only the most specific parameter matching your requirements.

For example, suppose that you have a rule with the *`Destination Zone`* parameter set, and you want to create a similar rule for a specific subnet of this zone. In this case, create a new rule with the *`Destination Subnet`* parameter set, do not set the *`Destination Zone`* parameter in both rules. Setting the *`Destination Zone`* parameter in both rules and setting the *`Destination Subnet`* parameter in the second rule would work for connections targeting the specified subnet, but it would cause Zorp to reject the connections that target other subnets of the specified destination zone, because both rules would match for the connection.

- The parameter groups are the following from the least specific to the most specific ones. Parameters within each group are listed from left to right from the least specific to the most specific ones.
  1. *`Destination Zone`* > *`Destination Interface Group`* > *`Destination Interface`* > *`Destination Subnet`*
  2. *`Source Zone`* > *`Source Subnet`*
  3. *`Destination Port`* (Note that port is more specific than port range.)
  4. *`Source Port`* (Note that port is more specific than port range.)
  5. *`Protocol`*
  6. *`Source Interface Group`* > *`Source Interface`* > *`VPN`*
- If no matching rule is found, Zorp rejects the connection.

Note
It is possible to create rules that are very similar, making debugging difficult.

6.5.2. Transparent and non-transparent traffic

Zorp can handle traffic both transparently and non-transparently. In transparent mode, the clients address the destination servers directly, without being aware that the traffic is handled by Zorp. In nontransparent mode, the clients address Zorp, that is, the destination address of the client's connection is an IP address of the firewall host, and Zorp determines the address of the destination server.

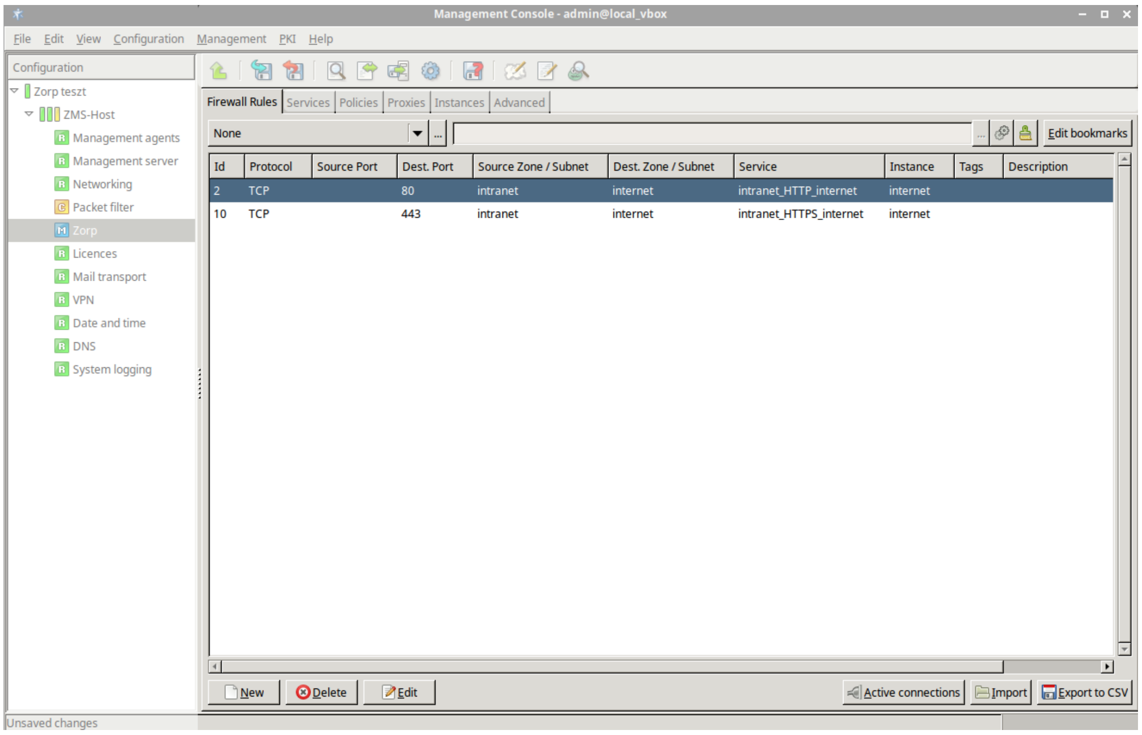
There are two methods to determine the destination address in nontransparent mode:

- **Inband destination selection**: Zorp can extract the destination address from the incoming traffic itself. This method is currently supported for HTTP and FTP connections. For example, see *Section 6.5.6, Configuring nontransparent rules with inband destination selection (p. 145)*.
- **Redirection**: Zorp can redirect the traffic to a fix destination address.

6.5.3. Procedure – Finding firewall rules

Step 1. Navigate to the **Firewall Rules** tab of the **Zorp** ZMC component.

Step 2. Use the filter bar above the rule list, to find rules. The use of the filter bar is described in *Section 3.3.10, Filtering list entries (p. 48)*. You can search for the parameters listed in *Section 6.5.1, Understanding Zorp firewall rules (p. 135)*.



Id	Protocol	Source Port	Dest. Port	Source Zone / Subnet	Dest. Zone / Subnet	Service	Instance	Tags	Description
2	TCP		80	intranet	internet	intranet_HTTP_internet	internet		
10	TCP		443	intranet	internet	intranet_HTTPS_internet	internet		

Figure 6.43. Finding rules

Step 3. Press **Enter**. The rules matching your search criteria are displayed.

Note
To save the list of matching rules into a file, click **Export to CSV**. Note that only the visible columns will be included in the file, in the order they are displayed.

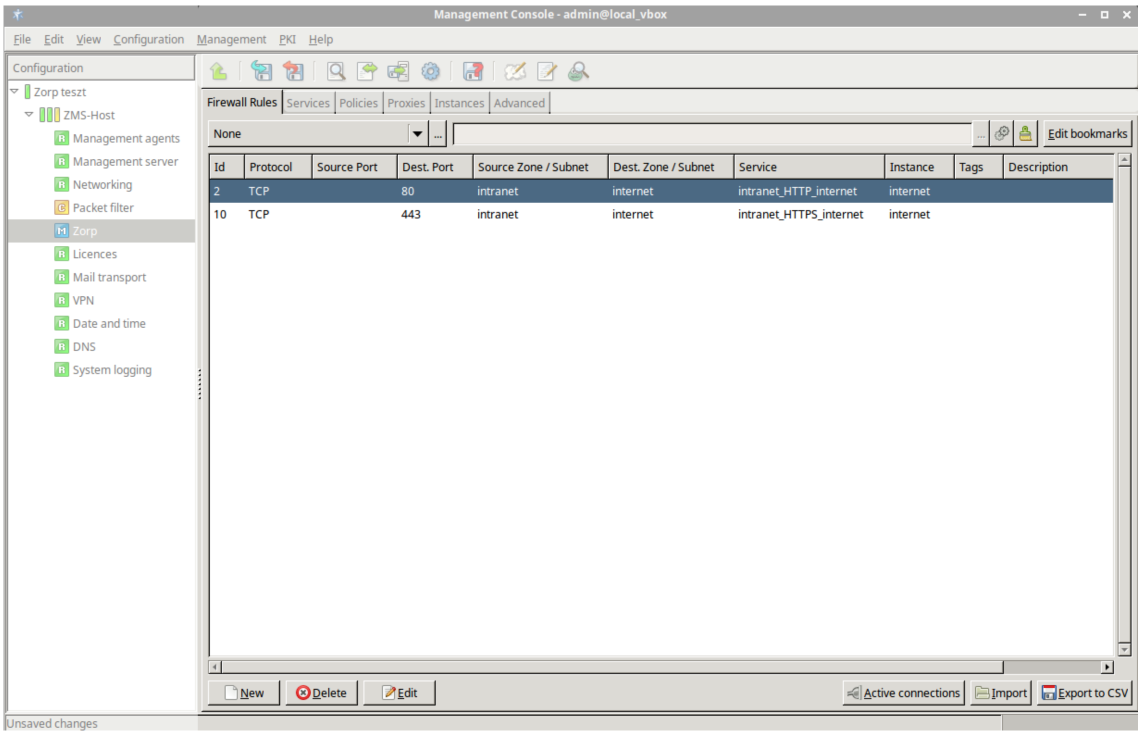
6.5.4. Procedure – Creating firewall rules

Purpose:

Firewall rules allow a specific type of traffic to pass the firewall. To create a new firewall rule, complete the following steps.

Steps:

Step 1. Login to ZMS and select **<Host>** > **Zorp** > **Firewall Rules** > **New**. A new window opens.



Id	Protocol	Source Port	Dest. Port	Source Zone / Subnet	Dest. Zone / Subnet	Service	Instance	Tags	Description
2	TCP		80	intranet	internet	intranet_HTTP_internet	internet		
10	TCP		443	intranet	internet	intranet_HTTPS_internet	internet		

Figure 6.44. Creating firewall rules

Step 2. Select the **Conditions** tab.

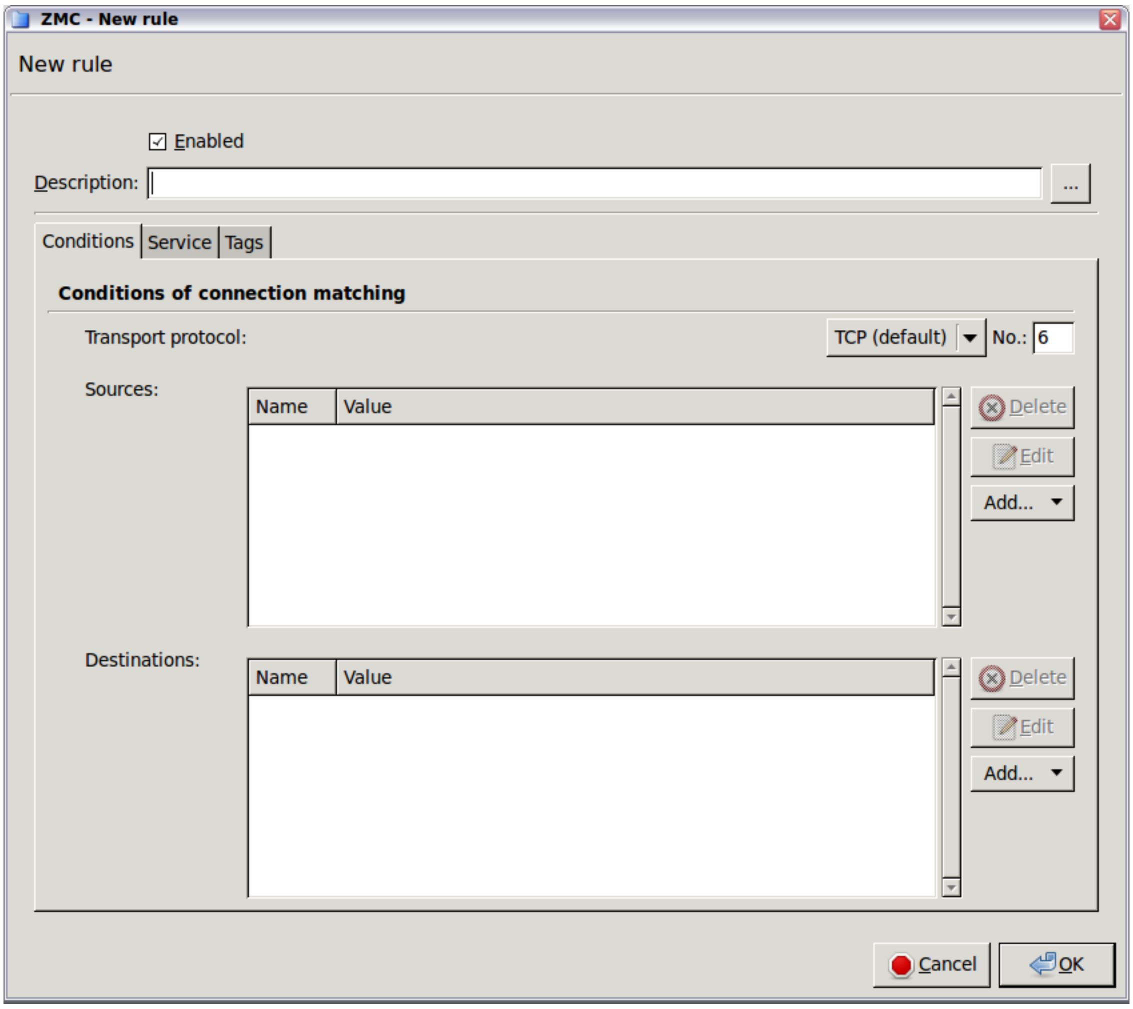


Figure 6.45. Setting connection parameters

Step 3. Select the **Transport protocol** used in the connection. This is the protocol used in the transport layer (Layer 4) of the OSI model. The following protocols are supported: *TCP*, *UDP*, *ICMP*, *IGMP*, *DCCP*, *GRE*, *ESP*, *AH*, *SCTP*, and *UDP-Lite*.

- To permit both TCP and UDP traffic, select **TCP or UDP**.
- To permit any Layer 4 protocol, select **Any**.
- For ICMP traffic, you can specify the permitted type and subtype (code) as well.

Step 4. Select the sources.
Zorp can limit the traffic that can pass the firewall only to traffic that comes from selected source networks. To permit traffic only from specific networks, select **Sources** > **Add** > **<Type-of-network>**. You can select zones, IPv4 or IPv6 subnets, interfaces, interface groups, and ports. Use always the most specific source suitable for your rule.

Note
To specify multiple ports, separate the ports with a comma, for example: *80,443*

To specify a port range, use a colon, for example: *2000:2100*

To specify multiple port ranges, separate the port ranges with commas, for example: *`2000:2100,2200:2400`*.

Step 5. Select the destinations.

Zorp can limit the traffic that can pass the firewall only to traffic that is targeting selected destination addresses. To permit traffic only to specific networks, select **Destinations** > **Add** > **<Type-of-network>**. You can select zones, IPv4 or IPv6 subnets, interfaces, interface groups, and ports. Use always the most specific destination suitable for your rule.

Note
For rules that start nontransparent services, set the destination address and the port to an address of the firewall host.

Note
To specify multiple ports, separate the ports with a comma, for example: *`80,443`*

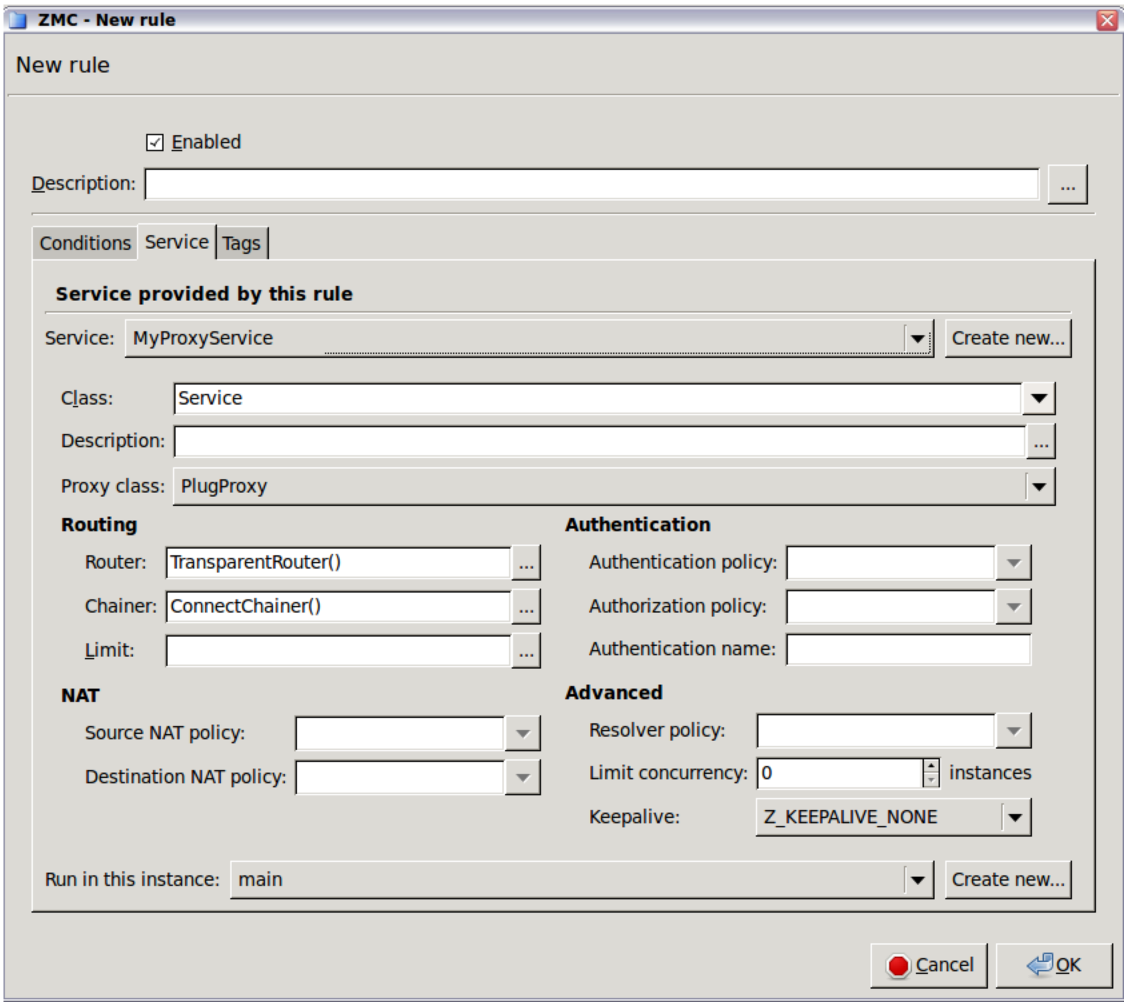
To specify a port range, use a colon, for example: *`2000:2100`*

To specify multiple port ranges, separate the port ranges with commas, for example: *`2000:2100,2200:2400`*.

Note
From Zorp version 3 F5 on, it is not mandatory to set the sources and destinations. Sources and destinations act as a filter, they limit access to the clients or servers of the sources and destinations. A firewall rule without sources and destinations acts as a rule that simply forwards traffic between any client and destination.

Step 6. Select the service to use.

Select **Service** > **Service** and select the service to start for connections matching the rule. The service determines the type of traffic that will be permitted by this rule (for example, HTTP, FTP, and so on) and also the level, the traffic will be inspected on (application or packet filter level).


ZMC - New rule

New rule

☑ Enabled

Description:

Conditions | Service | Tags

Service provided by this rule

Service: MyProxyService | Create new...

Class: Service

Description:

Proxy class: PlugProxy

Routing

Router: TransparentRouter()

Chainer: ConnectChainer()

Limit:

Authentication

Authentication policy:

Authorization policy:

Authentication name:

NAT

Source NAT policy:

Destination NAT policy:

Advanced

Resolver policy:

Limit concurrency: 0 instances

Keepalive: Z_KEEPALIVE_NONE

Run in this instance: main | Create new...

Cancel | OK

Figure 6.46. Selecting the service

Note
Proxy services can be used only if the **Condition** > **Transport protocol** option is set to *TCP*, *UDP*, or *TCP or UDP*.

Warning
The settings and parameters of the service shown on the **Service** tab of the rule are for reference only. Do not modify them, because it might interfere with other rules using the same service. To modify the parameters of a service, or to create a new service, use the **Services** tab of the **Zorp** ZMC component.

Step 7. Select the instance the service should run in.

Step 8. *Optional Step*: By default, new rules become active when the configuration is applied. To create a rule without activating it, deselect the **Active** option of the rule.

Step 9. *Optional Step*: To limit the number of connections that can be started by the rule, configure rate limits for the connections. For details, see *Procedure 6.5.7, Connection rate limiting (p. 146)*.

Step 10. Click **OK**, then commit your changes.

Expected result:

A new firewall rule is created and added to the list of firewall rules. If the rule is active, the traffic specified in the rule can pass the firewall.

6.5.5. Procedure – Tagging firewall rules

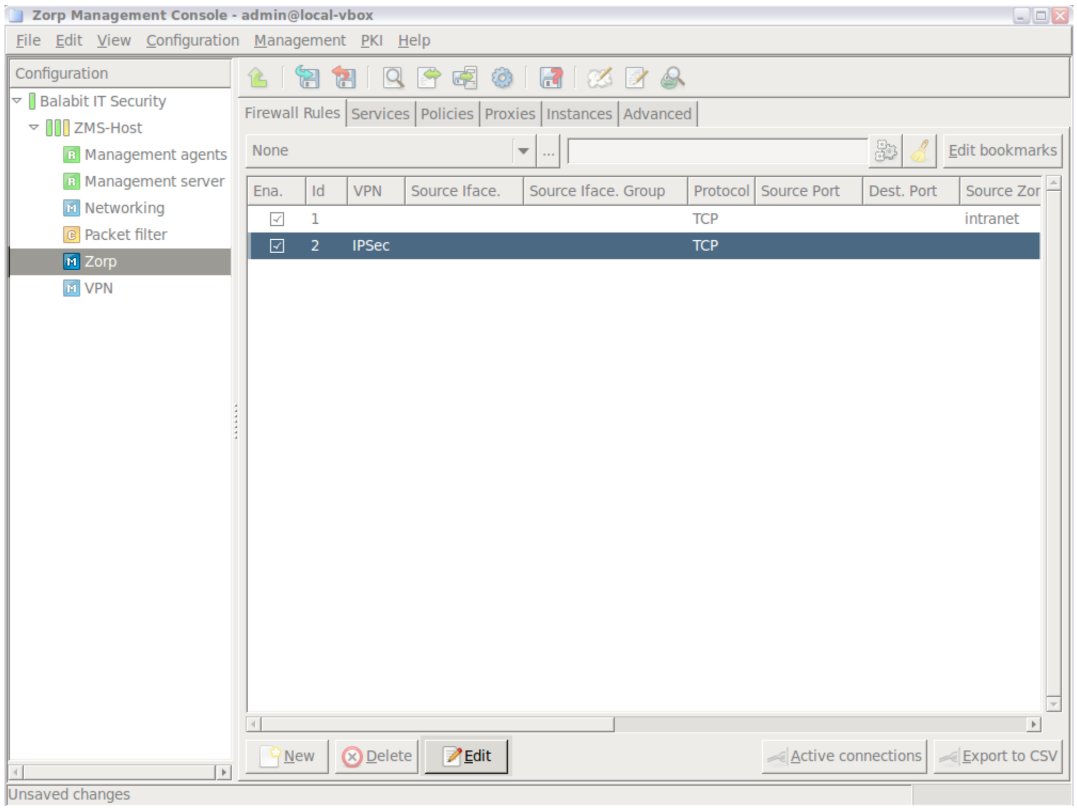
Purpose:

To add a tag to a firewall rule, complete the following steps. Tagging rules is useful, for example, to identify rules that belong to the same type of traffic.

Steps:

Step 1. Navigate to **<Host> > Zorp > Firewall Rules**.

Step 2. Select the rule to tag and click **Edit**.


Zorp Management Console - admin@local-vbox
File Edit View Configuration Management PKI Help
Configuration
Balabit IT Security
ZMS-Host
Management agents
Management server
Networking
Packet filter
Zorp
VPN
Firewall Rules Services Policies Proxies Instances Advanced
None
Edit bookmarks
New Delete Edit
Active connections Export to CSV
Unsaved changes

Ena.	Id	VPN	Source Iface.	Source Iface. Group	Protocol	Source Port	Dest. Port	Source Zor
☑	1				TCP			intranet
☑	2	IPSec			TCP			

Figure 6.47. Editing rules

Step 3. Click **Tags**. The list of available tags is displayed on the left; the tags assigned to the rule are displayed on the right.

- To create a new tag, click **New**, enter the name of the tag, then click **OK**.
- To assign a tag to the rule, select a tag and click **Assign**.

Note
Tags that are already assigned to the rule are not shown in the **Available tags** list.

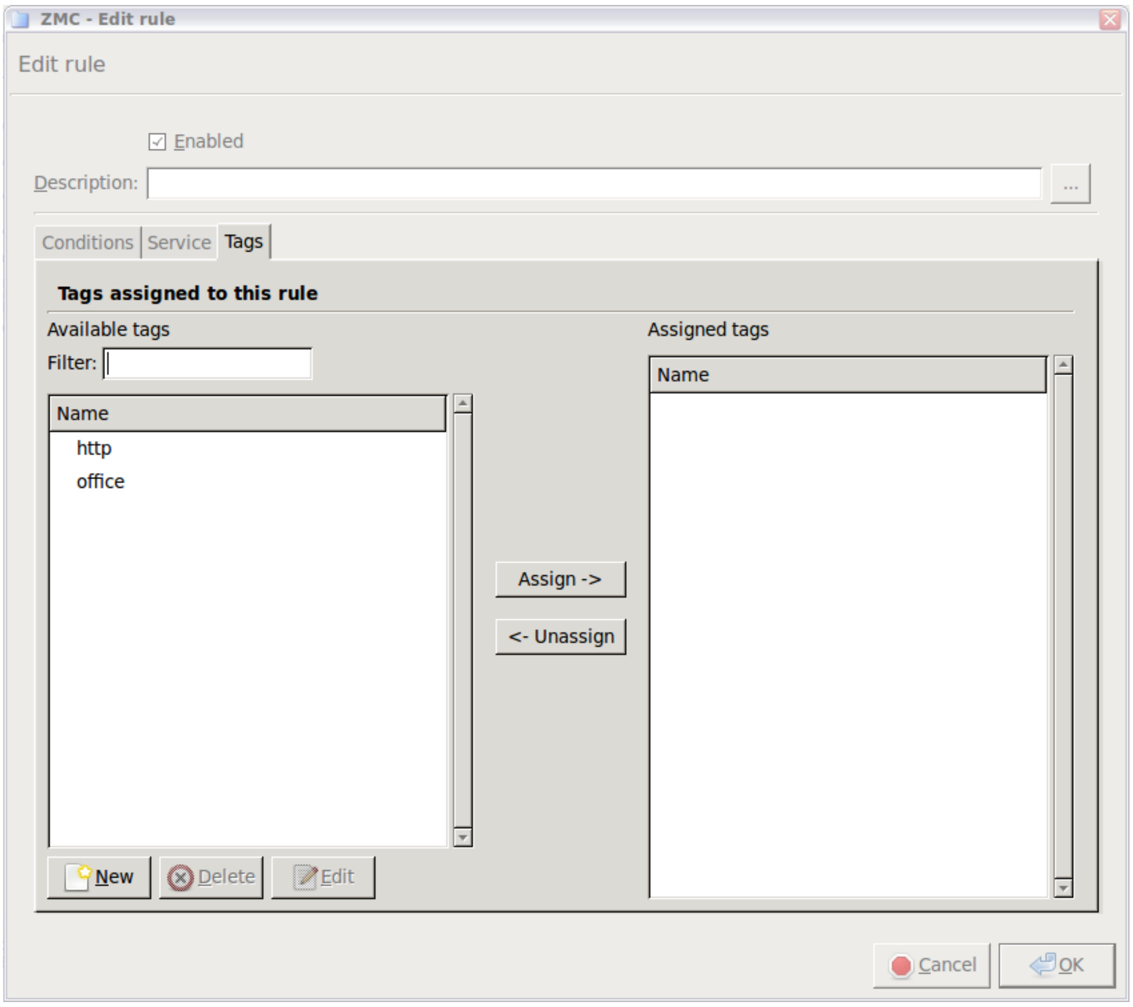


Figure 6.48. Tagging rules

Step 4. Click **OK**, then commit your changes.
Expected result:

The selected tags are assigned to the rule.

6.5.6. Configuring nontransparent rules with inband destination selection

When using inband destination selection, Zorp extracts the address of the destination server from the traffic. Note the following points:

- For HTTP connections, create a firewall rule that uses a nontransparent HTTP proxy and inband destination selection. Also, set the web browsers of the clients to use Zorp as a web proxy.

- If the clients use a caching web proxy for HTTP traffic, for example, Squid, and Zorp is located between the clients and the web proxy, then:
  - Create a firewall rule that uses a nontransparent HTTP proxy.
  - Set the *`parent_proxy`* and *`parent_proxy_port`* attributes of the proxy to the address of the caching proxy.
  - Use a *DirectedRouter* in the service to redirect the connections to the caching proxy, or use inband destination selection.

6.5.7. Procedure – Connection rate limiting

Purpose:

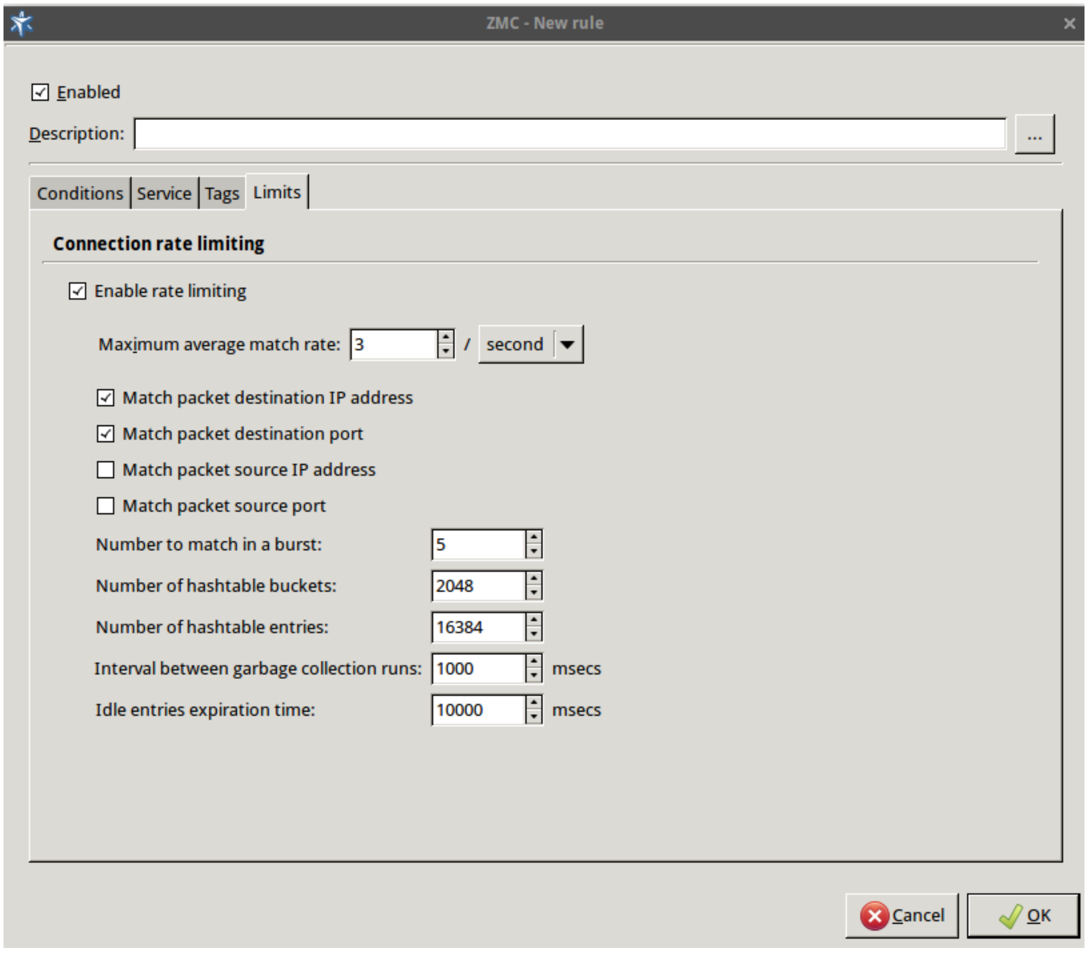
To limit the maximum rate of new connections in order to prevent from Denial of Service (DoS) attacks, configure the connection rate limiting options on the **Limits** tab of the firewall rule. You can specify the number of connections that Zorp accepts within a given time period. Connection requests above this maximum rate are denied.

Steps:

Step 1. Navigate to **<Host> > Zorp > Firewall Rules**.

Step 2. Select the rule to edit, then click **Edit > Limits**.

Step 3. Click **Enable rate limiting**, then set the maximum number of permitted connection requests (per second) in the **Maximum average match rate** field.



☑ Enabled

Description:

Connection rate limiting

☑ Enable rate limiting

Maximum average match rate: 3 / second

☑ Match packet destination IP address
☑ Match packet destination port
☐ Match packet source IP address
☐ Match packet source port

Number to match in a burst:	5	
Number of hashtable buckets:	2048	
Number of hashtable entries:	16384	
Interval between garbage collection runs:	1000	msecs
Idle entries expiration time:	10000	msecs

Figure 6.49. Connection rate limiting

Step 4.

- To limit the rate of connections based on the destination in the connection requests, select **Match packet destination IP address**.
- To limit the rate of connections based on the source of the connection requests, select **Match packet source IP address**.

Step 5. Set other parameters as needed for your environment.

6.6. Proxy classes

A proxy component is responsible for analyzing, filtering and possibly modifying the data that is passed through it.

Proxies work with data streams they receive as input and emit another (possibly altered) data stream as output. They never actually see “traffic” in the traditional sense; the details of connection establishment and management are handled by separate software components. Therefore, proxies can easily interoperate with each other or with other non-firewall software like virus filters, since data is passed among them as simple data stream.

A proxy class is the low level proxy and its configuration settings. Proxy classes are responsible for analyzing and checking the data part of packets and passing it between the client and the server. Proxy classes can be used in service definitions. Together with the other components configured as service parameters (for example, routers) they can be used to analyze/filter communication channels among network hosts. Most proxy classes are protocol-specific, meaning that they are aware of the protocol-specific information within the data stream they are processing.

There are built-in proxy classes for the most typical network traffic types, like HTTP, FTP, POP3, SMTP, telnet, and also for some less frequently used protocols, like NNTP and LDAP. They can be used in service definitions without modifications of the default class properties and their values. See the *Zorp Professional 7 Reference Guide* for details.

Note
Proxies in Zorp are fully RFC-compliant so a traffic that pretends to be of a certain type but violates RFC specifications concerning the given traffic type are not proxied but automatically denied instead. For example, the HTTP proxy enforces by default the RFCs for HTTP (2616 and 1945).

Example 6.7. RFC-compliant proxying in Zorp
A good example for this rule is the CODE RED worm that infected so many IIS servers around the world: the heart of this worm was a specially formatted URL request which was not RFC-compliant but was nevertheless serviced by IIS servers that had not been patched against it. Most firewall products, even application proxy firewalls let it pass through and only the most accurate ones, like Zorp, stopped the worm, realizing that the URL request coming from the worm violated RFC rules.

If using default proxy classes and property values is not enough, it is possible to derive new classes from the original ones. Derived proxy classes inherit all the properties of the original (parent) classes and these properties can then be altered. The number of configurable parameters varies among proxies; proxy for HTTP traffic has the most. It is completely up to the administrator whether and to what extent they are used in the firewall's policy settings.

Note
Whenever you start customizing a proxy, you do not actually create a new proxy, but derive a proxy class from the selected built-in proxy implementation and configure different settings for it. You only modify the configuration, not the proxy module itself.

6.6.1. Customizing proxies

Default proxy classes provide an adequate level of security. Own proxy classes are typically derived from these default proxy classes in case there is need to change the values of certain attributes, like, for example, manually setting the content of the request headers leaving the HTTP proxy (like browser type, operating system). Complex proxy setups – such as virus filtering of HTTP, SMTP, traffic or proxy stacking – also require derived classes.

This process is somewhat complex involving many steps; therefore it will be demonstrated using an example of changing the `User-Agent` HTTP request header output by a custom HTTP proxy component.

The customized proxy class you are defining is based on an already defined proxy class. There are quite a lot of predefined proxy classes that are available by default. For some protocols (for example HTTP and FTP) there are more than one to choose from, each with a specific intended purpose. `FtpProxyRO`, for instance, is for read-only FTP access, while `FtpProxyRW` is for read/write FTP access.

6.6.1.1. Procedure – Derive a new proxy class

Step 1. Select the **Proxies** tab of the `Zorp` component.
The Zorp class configuration window that appears is empty by default.

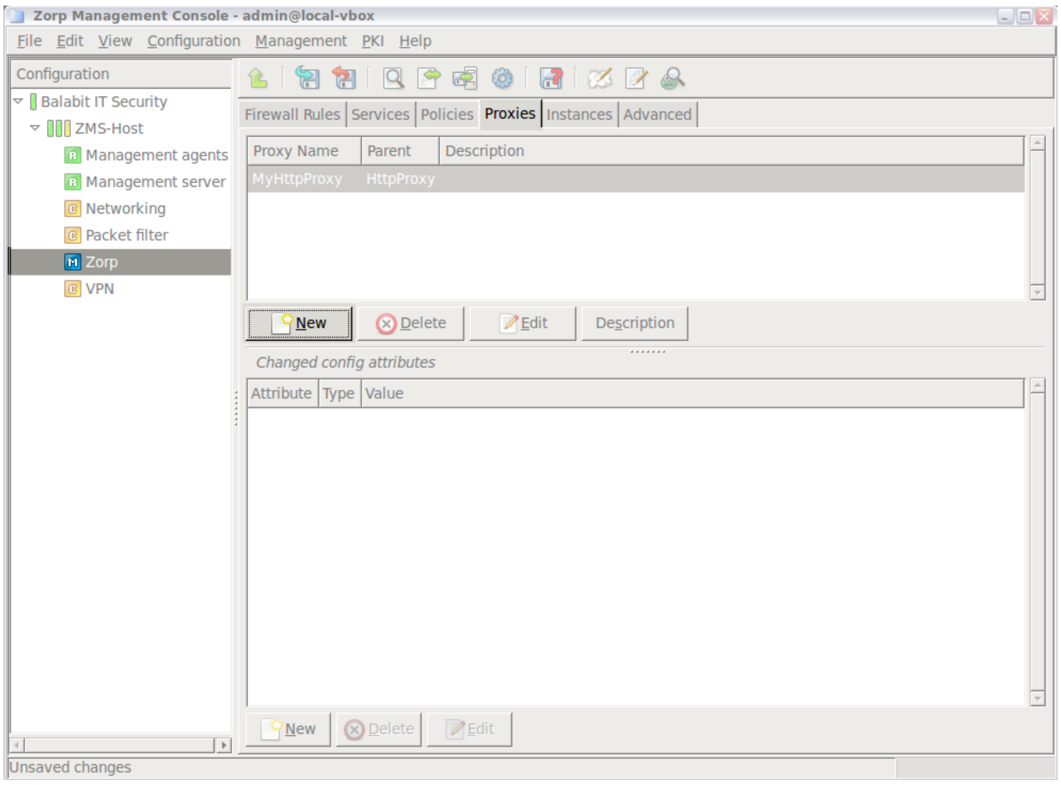
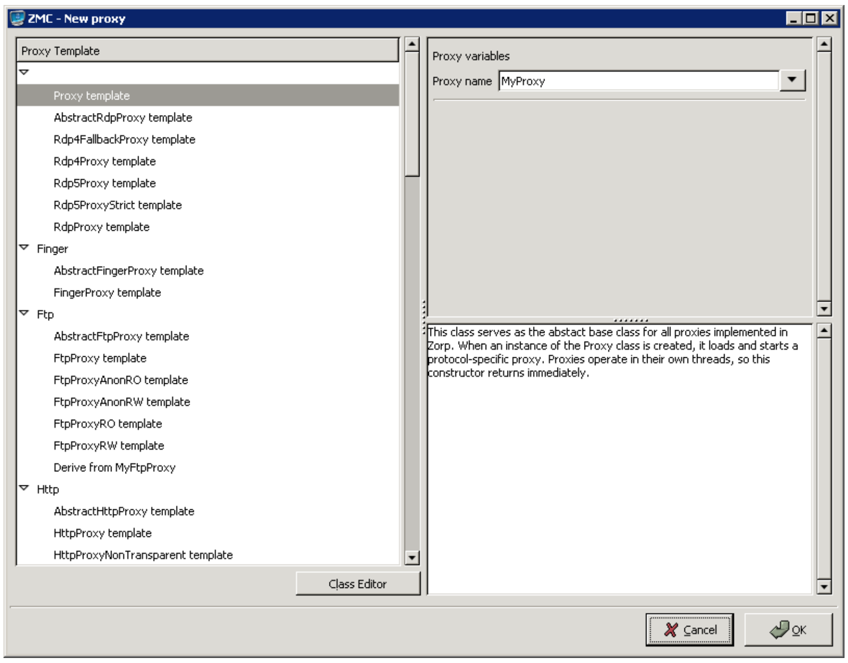


Figure 6.50. Deriving new proxy classes

Step 2. Click **New**.

Step 3. Select a predefined proxy class template for the customized proxy class.


ZMC - New proxy

Proxy Template

Proxy template
AbstractRdpProxy template
Rdp4FallbackProxy template
Rdp4Proxy template
Rdp5Proxy template
Rdp5ProxyStrict template
RdpProxy template
Finger
AbstractFingerProxy template
FingerProxy template
Ftp
AbstractFtpProxy template
FtpProxy template
FtpProxyAnonRO template
FtpProxyAnonRW template
FtpProxyRO template
FtpProxyRW template
Derive from MyFtpProxy
Http
AbstractHttpProxy template
HttpProxy template
HttpProxyNonTransparent template

Proxy variables
Proxy name: MyProxy

This class serves as the abstract base class for all proxies implemented in Zorp. When an instance of the Proxy class is created, it loads and starts a protocol-specific proxy. Proxies operate in their own threads, so this constructor returns immediately.

Figure 6.51. Default proxy templates

Proxy class names are typically descriptive, but most of them come with a detailed description as well. For more details, see the *Zorp Professional 7 Reference Guide*. These descriptions either explicitly tell what the given proxy class is for, or suggest attributes of the class that can be configured to achieve a special purpose.

Step 4. Enter a name for the new proxy. It is recommended to use capital names that imply the functions the proxy is responsible for, for example `VirusHTTP` or `HTTPSproxy`.

Step 5. Add the attributes to be configured and modify attribute values for the given class. For details, see *Procedure 6.6.1.2, Customizing proxy attributes (p. 150)*.

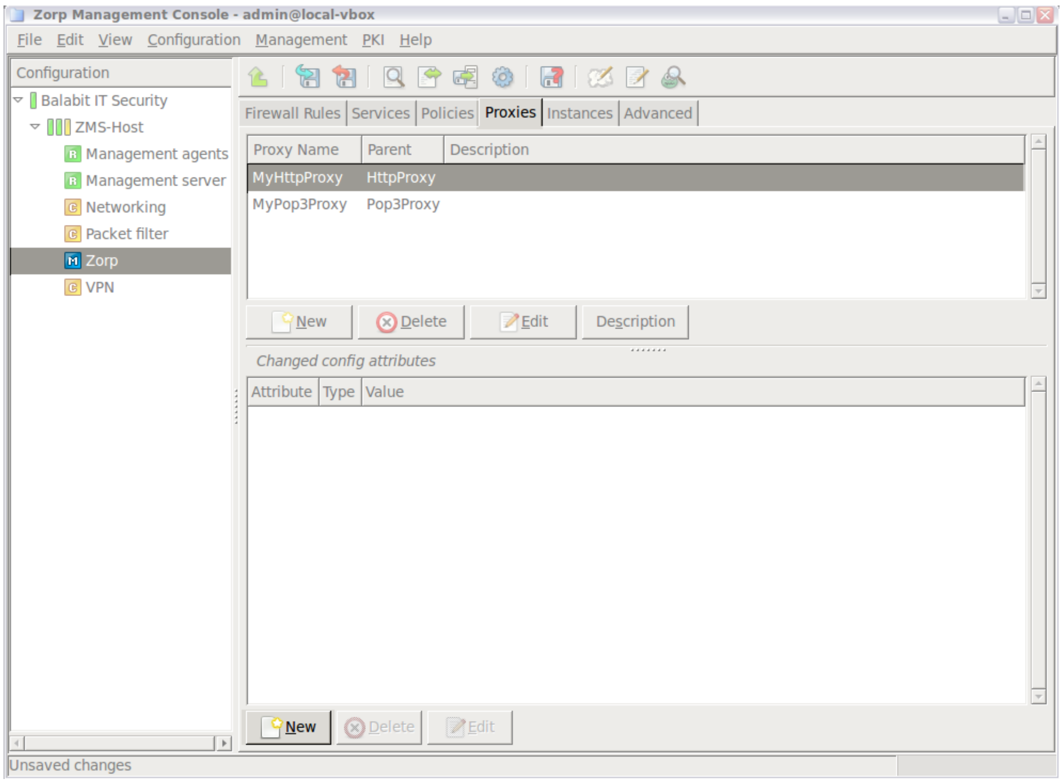
Note
To have the new proxy class fully functional, you have to configure it as a service parameter of the given service.

6.6.1.2. Procedure – Customizing proxy attributes

What attribute-level configuration is needed depends on the exact requirements: if you simply need an FTP proxy that denies upload (write) requests, use the `FtpProxyRO` without modifications in your policy definitions – deriving a new class is unnecessary in this case.

However, if you would like to hide the browser type and operating system version information of your clients you can do it with a derived proxy class, by customizing some of its attributes. To hide browser type and operating system version information for instance, the creation of a custom `User-Agent` header is required. Although this may be accomplished on the client side (modifying all client web browsers), it is much easier to do with Zorp.

The attributes configuration screen is divided into two main parts.



Proxy Name	Parent	Description
MyHttpProxy	HttpProxy	
MyPop3Proxy	Pop3Proxy	

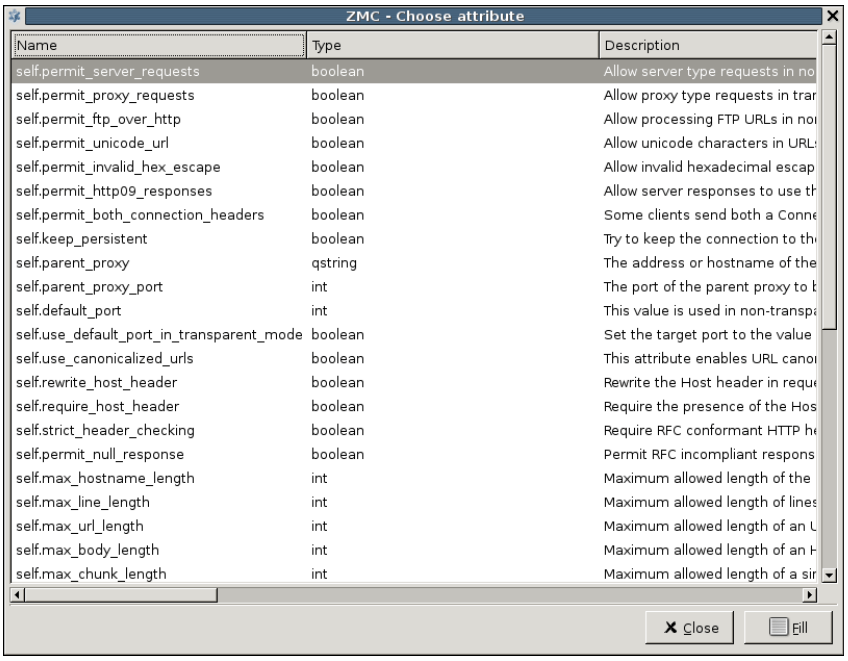
Attribute	Type	Value

Figure 6.52. Customizing proxy attributes

The upper textbox shows the list of custom, derived proxies along with the classes they were derived from (the Parent column). For the previous screenshot a simple HttpProxy, called `MyHttpProxy` was derived from the generic HttpProxy class.

Step 1. Navigate to **Zorp** > **Proxies** and select the proxy to customize.

Step 2. Click **New** under the lower table. The list of configurable attributes are displayed.



Name	Type	Description
self.permit_server_requests	boolean	Allow server type requests in no
self.permit_proxy_requests	boolean	Allow proxy type requests in tra
self.permit_ftp_over_http	boolean	Allow processing FTP URLs in no
self.permit_unicode_url	boolean	Allow unicode characters in URL
self.permit_invalid_hex_escape	boolean	Allow invalid hexadecimal escap
self.permit_http09_responses	boolean	Allow server responses to use th
self.permit_both_connection_headers	boolean	Some clients send both a Conne
self.keep_persistent	boolean	Try to keep the connection to th
self.parent_proxy	qstring	The address or hostname of the
self.parent_proxy_port	int	The port of the parent proxy to
self.default_port	int	This value is used in non-transp
self.use_default_port_in_transparent_mode	boolean	Set the target port to the value
self.use_canonicalized_urls	boolean	This attribute enables URL cano
self.rewrite_host_header	boolean	Rewrite the Host header in reque
self.require_host_header	boolean	Require the presence of the Hos
self.strict_header_checking	boolean	Require RFC conformant HTTP he
self.permit_null_response	boolean	Permit RFC incompliant respons
self.max_hostname_length	int	Maximum allowed length of the
self.max_line_length	int	Maximum allowed length of lines
self.max_url_length	int	Maximum allowed length of an
self.max_body_length	int	Maximum allowed length of an
self.max_chunk_length	int	Maximum allowed length of a si

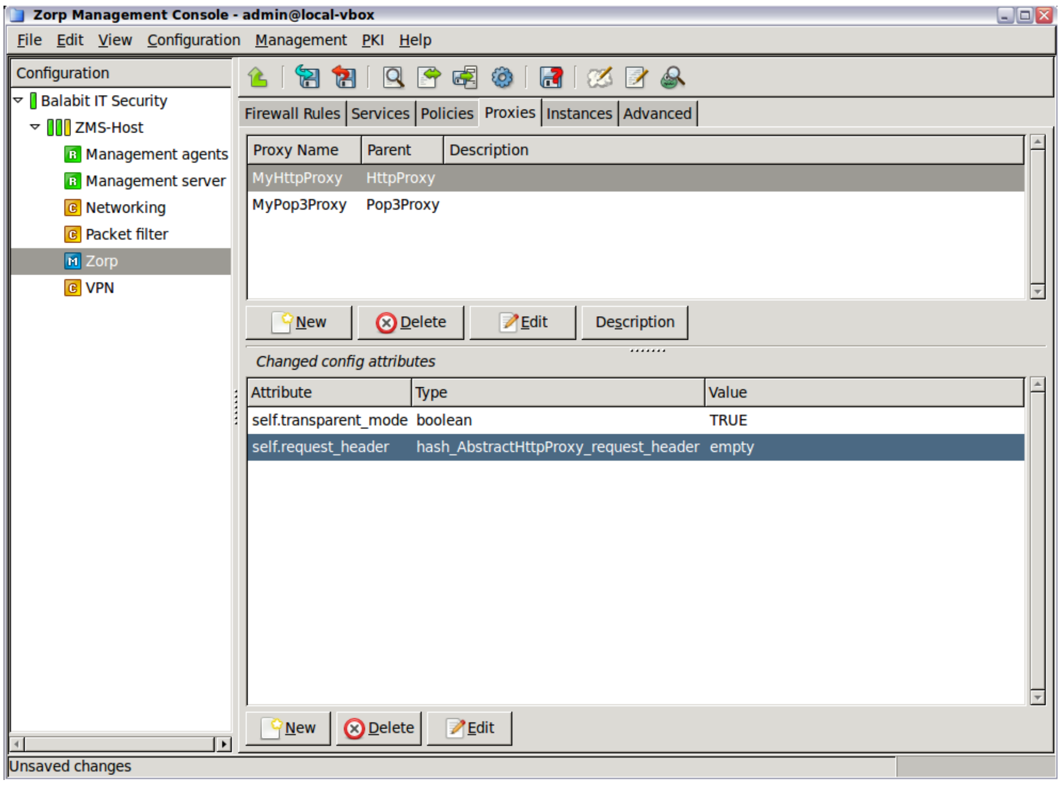
Figure 6.53. Listing of proxy class attributes

Note
A short description for each attribute is also displayed. For a complete description of proxy classes and attributes see the *Zorp Professional 7 Reference Guide*.

There are syntax rules for setting attributes properly. For more information on these rules, see the *Zorp Professional 7 Reference Guide* or, to a limited extent, read all the available descriptions on the class selection screen.

Tip
AbstractProxy template descriptions are especially useful, since they contain the most information on syntax. For example, to set HTTP request headers in the traffic, see *Section 4.7.2.2, Configuring policies for HTTP requests and responses* in *Zorp Professional 7 Reference Guide*.

Step 3. Select `self.request_header` attribute.
The attribute appears in the **Changed proxy attributes** listing of the Zorp class configuration screen.

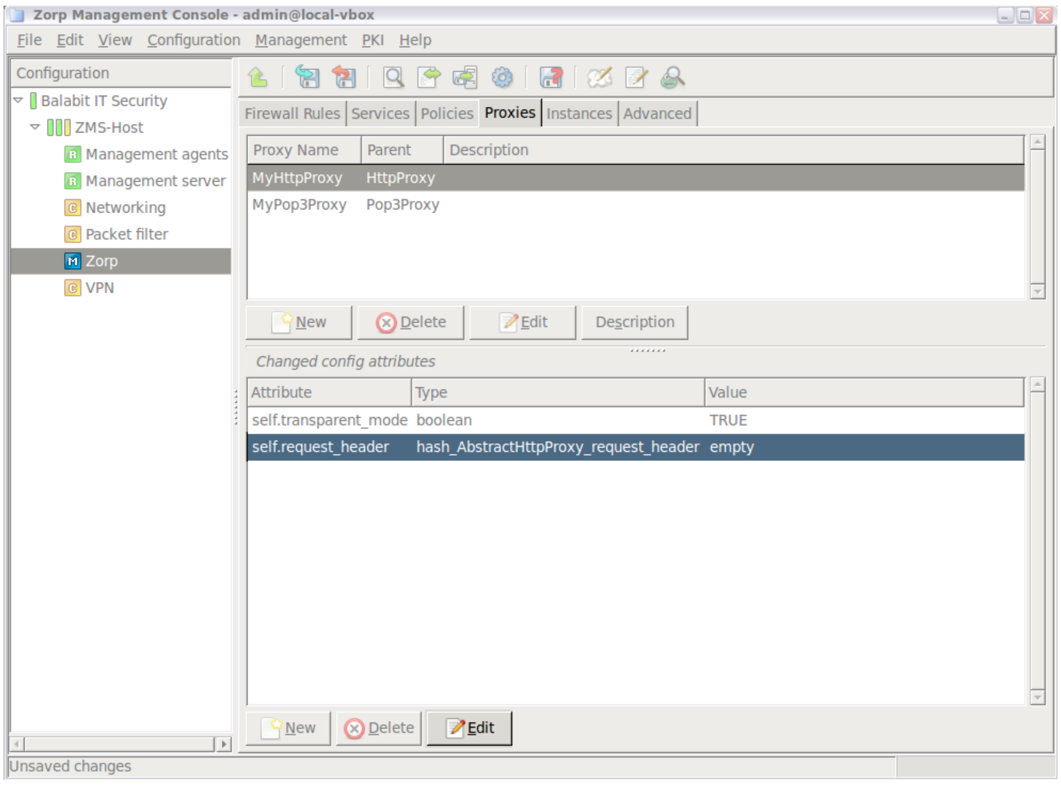


Proxy Name	Parent	Description
MyHttpProxy	HttpProxy	
MyPop3Proxy	Pop3Proxy	

Attribute	Type	Value
self.transparent_mode	boolean	TRUE
self.request_header	hash_AbstractHttpProxy_request_header	empty

Figure 6.54. The newly added self-request_header attribute

Step 4. Set the value of the attribute by clicking **Edit**. (The attribute Type is less relevant now.) A new window opens which is, by default, empty.



Proxy Name	Parent	Description
MyHttpProxy	HttpProxy	
MyPop3Proxy	Pop3Proxy	

Attribute	Type	Value
self.transparent_mode	boolean	TRUE
self.request_header	hash_AbstractHttpProxy_request_header	empty

Figure 6.55. Editing an attribute

Step 5. Click the **New** button to define the name of the parameter you want to change.

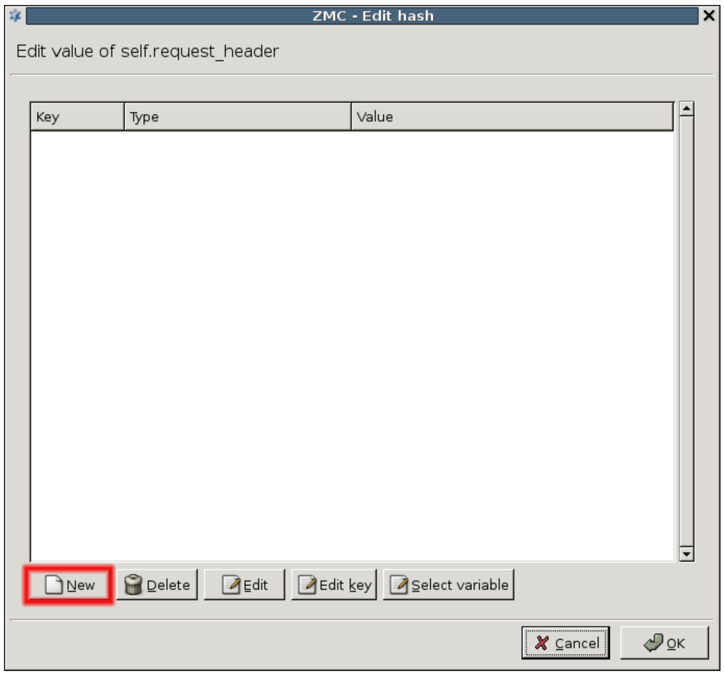


Figure 6.56. Modifying an attribute

In this example HTTP request headers are configured. These are standardized in the corresponding RFC documentation or in any studies or literature on web server administration/programming.

One of the request headers is called `User-Agent` which is the place to specify the browser type, version and operating system information. Popular statistics, such as the market share of web browsers, are based on this request header.

By default, Zorp takes the original `User-Agent` header information it receives from clients and uses the same value in HTTP requests it generates.

Step 6. Enter `User-Agent` into the small dialog box to change the default behavior.
You can see the name of the header changing (Key column), but the Type and Value columns still need to be changed.

Step 7. Left-click on the **Type** column of the row containing the previously entered *`User-Agent`* string, a drop-down list appears. In order to change the value of an existing attribute, select the `type_http_hdr_change_value` here, which changes the given header values.

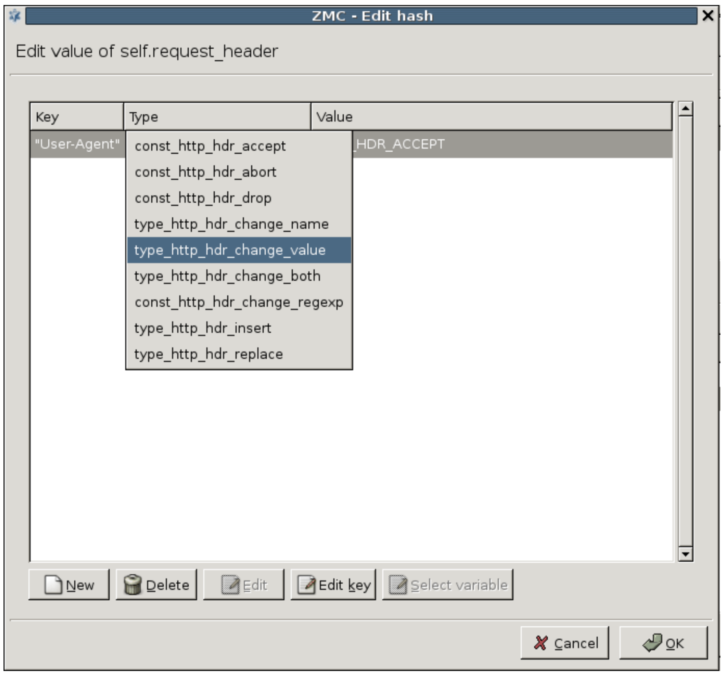
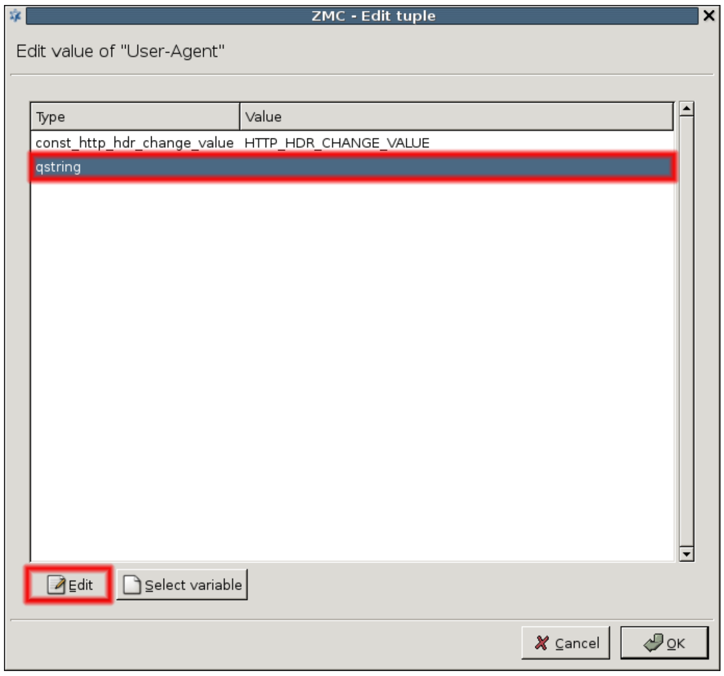


Figure 6.57. Selecting action type for the attribute

Step 8. Click **Edit** to modify the Value column.
Set the actual value of the `User-Agent` request header. The following window opens.


ZMC - Edit tuple

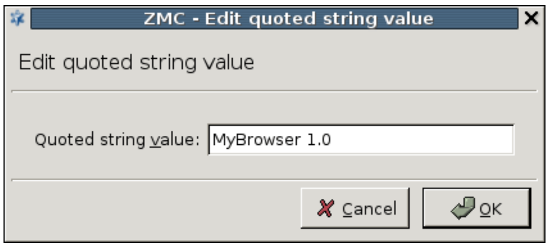
Edit value of "User-Agent"

Type	Value
const_http_hdr_change_value	HTTP_HDR_CHANGE_VALUE
qstring	

Figure 6.58. Editing the value of the User-Agent header

This window presents another view of the attribute you are modifying now. The Type column of Figure *Selecting action type for the attribute* is now the first row in this window, while the Value column became the second row here; it is currently empty.

Step 9. Click **Edit** to set the Value column and enter a string.


ZMC - Edit quoted string value

Edit quoted string value

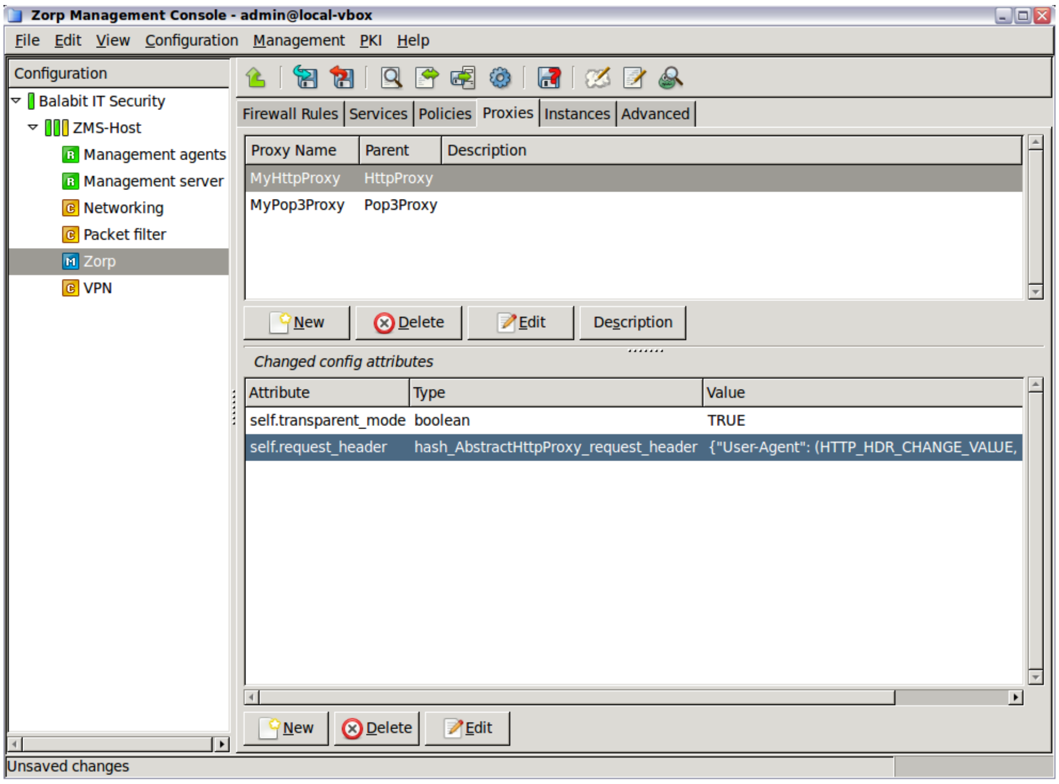
Quoted string value: MyBrowser 1.0

Figure 6.59. Editing the User-Agent header

A string can be for example, `My Browser`.

Note
The web servers you visit from now on will see this information as the `User-Agent` header they receive, and may act strangely if they, or the content they serve (Java Servlets, for instance) are not prepared to handle unexpected values in `User-Agent` headers.

Step 10. The process of changing the desired proxy class attribute is complete, you can see the result in the Zorp class configuration window.


Zorp Management Console - admin@local-vbox
File Edit View Configuration Management PKI Help
Configuration
Balabit IT Security
ZMS-Host
Management agents
Management server
Networking
Packet filter
Zorp
VPN
Firewall Rules Services Policies Proxies Instances Advanced
New Delete Edit Description
Changed config attributes
New Delete Edit
Unsaved changes

Proxy Name	Parent	Description
MyHttpProxy	HttpProxy	
MyPop3Proxy	Pop3Proxy	

Attribute	Type	Value
self.transparent_mode	boolean	TRUE
self.request_header	hash_AbstractHttpProxy_request_header	{"User-Agent": (HTTP_HDR_CHANGE_VALUE,

Figure 6.60. The User-Agent request header attribute is changed

6.6.1.3. Customized proxies and the services

The **proxy class** parameter configured in the service definition defines what traffic passes through with the help of the given service. Different services can have different **proxy class**es configured, even for the same traffic type. For example, a firewall can have a number of services configured to pass HTTP traffic, each with its own, derived and customized proxy class parameter. If these **proxy class**es are parameterized differently, the corresponding services also behave differently for the same HTTP traffic. A single service can only have a single **proxy class** value configured, although that proxy class parameter can refer to a stacked proxy setup as well.

6.6.2. Procedure – Renaming and editing proxy classes

To modify the basic attributes, for example, the name and the parent class of a proxy, complete the following steps:

Step 1. Select the class and click **Edit**.

Step 2. To rename the class, edit the name of the proxy in the **Proxy name** field, then click **Ok**.

Step 3. To modify the parent class of the proxy, select the new class from the **Proxy template** tree on the left of the dialog, then click **Ok**.

Warning
As a result of modifying the parent class, an already configured proxy (that is, one that had its default values or attributes modified) looses the attributes not present in its new parent class.

6.6.3. Analyzing embedded traffic

Most Zorp proxies can pass the information received as the payload of the incoming traffic to another proxy for further analysis. This kind of complex data analysis is possible by placing a proxy inside another one. This process is called stacking. Stacking is especially useful in filtering compound traffic, a traffic that consists of two (or more) protocols or that needs to be analyzed in two different ways.

Note
Earlier versions of Zorp used stacking to inspect encrypted protocols, for example, HTTPS or IMAPS. Now every proxy can decrypt SSL and TLS encryption without having to use another proxy. For details on configuring Zorp to handle encrypted connections, see *How to configure SSL proxying in Zorp 7*.

Usually protocols consist of two parts:

- control information, and
- data.

Protocol proxies analyze and filter the control part and except for some cases they are unaware of the data part. At this point, further screening of the data might be needed, therefore proxies are able to stack in other proxies capable of filtering the data part, so the external (upper) proxy passes that data traffic to the internal (lower) proxy.

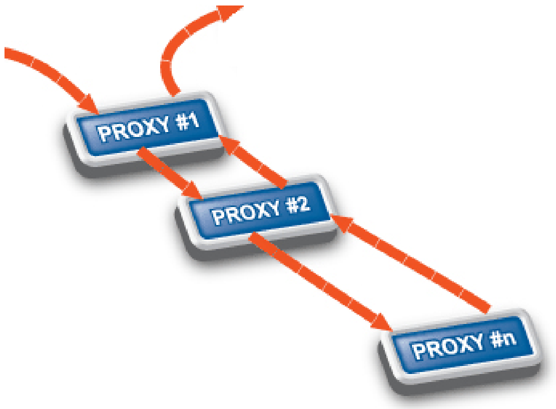


Figure 6.61. Stacking proxies

Example 6.8. Virus filtering and stacked proxies
Virus filtering is also part of the multiple analysis on traffic. It is typically performed in HTTP, POP3 and SMTP traffic, because these are the protocols, viruses generally use for spreading over the Internet (using Zorp though, it is possible to filter viruses in other protocols as well). When virus filtering is configured, a standard protocol proxy works in tandem with an antivirus engine and this way, both protocol-specific filtering and virus filtering are performed on the data if you stack the antivirus engine into some proxy.

For details on configuring virus filtering in HTTP and HTTPS traffic, see *How to configure virus filtering in HTTP*.

For each stacking scenarios there are a number of attributes that can be configured. For more information see the *Zorp Professional 7 Reference Guide*.

6.6.3.1. Procedure – Stack proxies

Since Zorp proxies natively support TLS-encrypted connections, stacking proxies is rarely needed in Zorp. For details on configuring TLS-encrypted connections (for example, HTTPS), see *How to configure SSL proxying in Zorp 7*. If you need to stack proxies for some reason, complete the following steps.

Step 1. Derive a proxy class from one of the predefined ones. For details, see *Section 6.6.1, Customizing proxies (p. 148)*.

Step 2. Derive a proxy class for the 'external' or parent proxy.

Step 3. Configure stacking on the Zorp Class configuration screen.
Configuring stacking essentially means setting an attribute of the container proxy. The exact name of the attribute depends on the parent proxy. For details, see the *Zorp Professional 7 Reference Guide*. For details on setting proxy attributes, see *Procedure 6.6.1.2, Customizing proxy attributes (p. 150)*.

6.7. Policies

The **Policies** tab provides a single interface for managing all the different policies used in Zorp service definitions.

Policies are independent from service definitions. A single policy can be used in several services or proxy classes. Policies must be created and properly configured before they are actually used in a service. When configuring a service, only the existing policies, that is, the previously defined ones, can be selected.

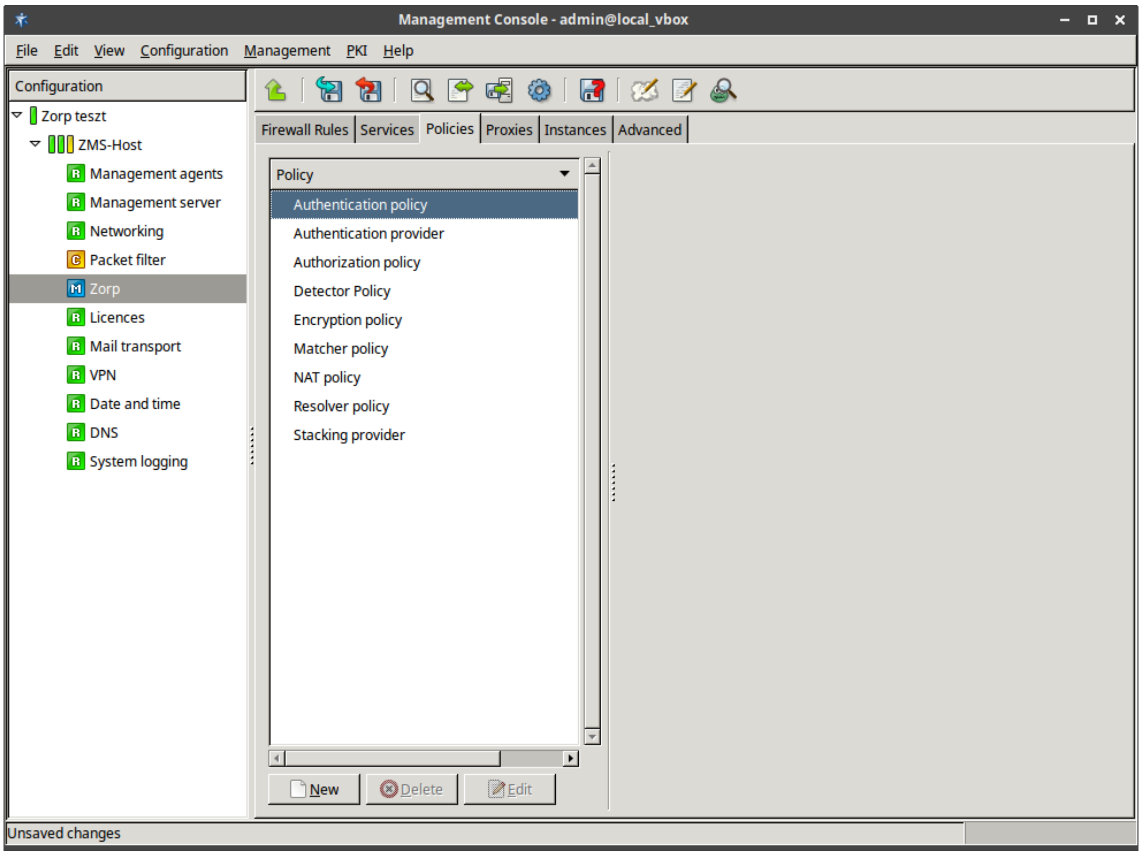


Figure 6.62. The Policies tab

On the left side of the Policies tab the existing policies are displayed in a tree, sorted by policy type. If a policy is selected, its parameters are displayed on the right side of the panel.

The policies available from the **Policies** tab of the ZMC **Zorp** component are listed below. The subsequent sections describe the different policy types and their uses. The Authentication, Authentication Provider, and Authorization policies are discussed in *Chapter 15, Connection authentication and authorization (p. 389)*.

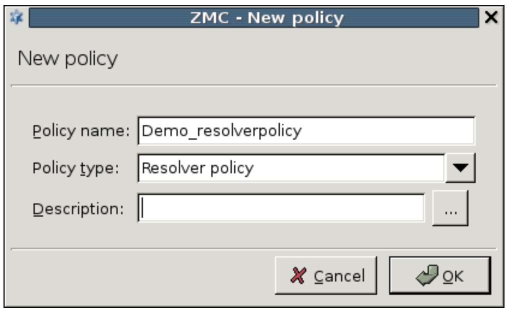
- *NAT Policy*: NAT policies are created for source and destination network address translation.
- *Authentication Policy*: These policies describe how clients should be authenticated in different scenarios. See *Chapter 15, Connection authentication and authorization (p. 389)* for details.
- *Authentication Provider*: ZAS authentication backends: The databases provide user information for authentication. See *Chapter 15, Connection authentication and authorization (p. 389)* for details.
- *Authorization Policy*: These policies describe how clients should be authorized in different scenarios. See *Chapter 15, Connection authentication and authorization (p. 389)* for details.
- *Detector Policy*: These policies start a service based on the protocol used in the connection.
- *Matcher Policy*: These are various matcher and filtering policies.
- *Resolver Policy*: These policies are simple domain name resolution policies.
- *Stacking Provider*: The stacking provider is an external Zorp host that provide content vectoring capabilities.

6.7.1. Procedure – Creating and managing policies

To create a new policy, complete the following steps.

Step 1. Navigate to **Zorp** > **Policies** and click the **New** button under the policies list.

Step 2. Enter a name for the policy and select the type of the new policy from **Policy type** combobox. If a policy or a policy group (for example, NAT policy) is selected, the combobox is automatically set to the same type.



Policy name: Demo_resolverpolicy

Policy type: Resolver policy

Description:

Figure 6.63. Defining new policies

Note
Custom policy classes can be created using the **Class editor**. However, this is only recommended for advanced users.

Step 3. Configure the parameters of the new class on the right side of the panel. Existing policy classes can be modified here as well.

Tip

- To disable a policy or a matcher, use the local menu (right-click the selected policy). Disabled policies will be generated into the configuration files as comments.
- To duplicate a policy, use the **Copy** and **Paste** options of the local or the **Edit** menu.

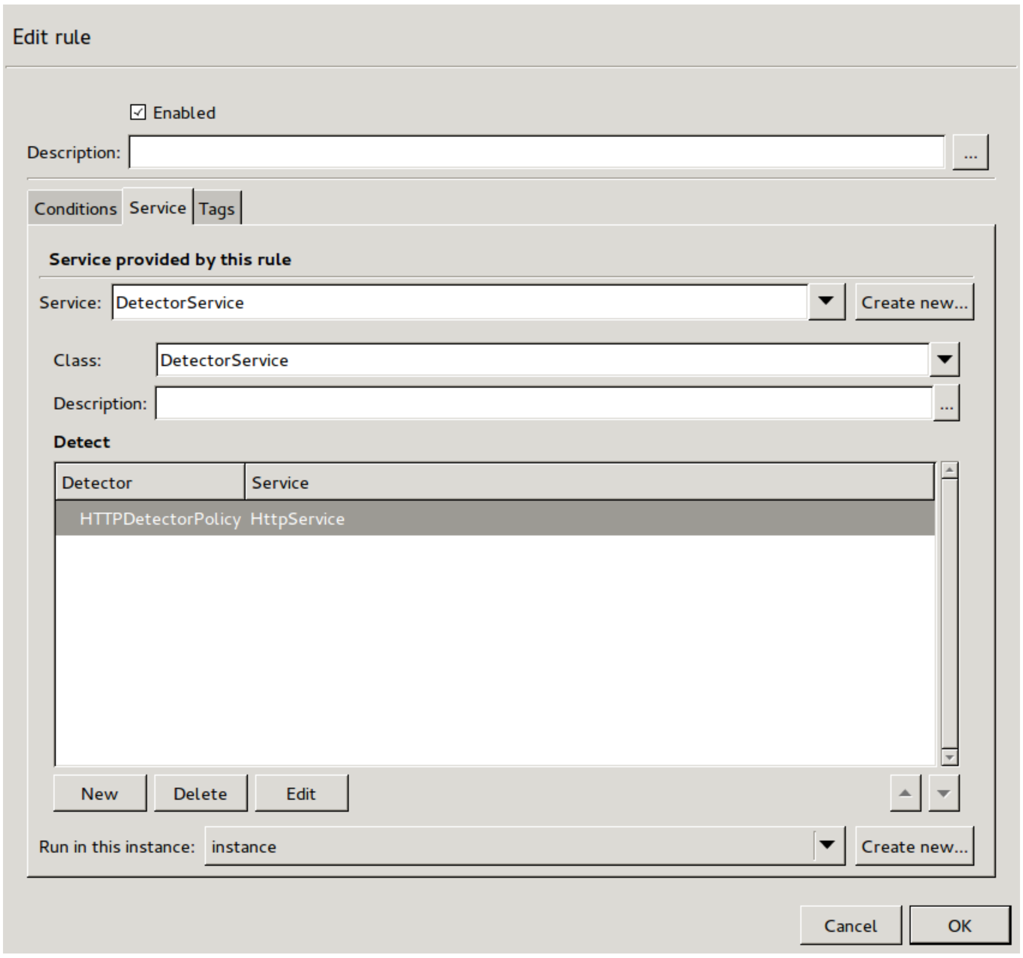
6.7.2. Detector policies

Detector policies can be used in firewall rules with DetectorServices: they specify which service should Zorp start for a specific type of protocol or certificate found in the traffic of connection. Currently Detector policies cam detect the HTTP, SSH, and SSL protocols. In SSL/TLS-encrypted connections, Zorp can select which service to start based on the certificate of the server.

For example, you can use this feature for the following purposes:

- to enable HTTP traffic on non-standard ports
- to enable SSL-encrypted traffic only to specific servers or server farms that do not have public IP addresses (for example, Microsoft Windows Update servers)

- to enable access only to specific HTTPS services (for example, enable access to Google Search (which has a certificate issued to `www.google.com`), but deny access to GMail (which uses a certificate issued to `accounts.google.com`))



Service: DetectorService

Class: DetectorService

Detector	Service
HTTPDetectorPolicy	HttpService

Run in this instance: instance

Figure 6.64. Detector policies

Detector policies contain a detector that determines if the traffic in a connection belongs to a particular protocol or not. Firewall rules can include a list of detector-service pairs. When a client opens a connection that is handled by a DetectorService, Zorp evaluates the detectors in the Detector policy of the DetectorService, starting with the first detector. When a detector matches the traffic in the connection, Zorp starts the service set for the detector to handle the connection. If none of the detectors match the traffic, then Zorp terminates the connection.

Example 6.9. Defining a Detector policy

Python: Below is a Detector policy that detects SSH traffic in a connection.

```
DetectorPolicy(name="DemoSSHDetector", detector=SshDetector())
DetectorPolicy(name="DemoHTTPDetector", detector=HttpDetector()
```

The following firewall rule uses the DemoSSHDetector Detector policy to start the DemoSSHService service if SSH traffic is found in a connection, and drops other connections.

```
Rule(rule_id=1,
    proto=6,
    detect={'DemoSSHDetector' : 'demo_instance/DemoSSHService',}
    )
```

The following firewall rule uses the DemoSSHDetector and the DemoHTTPDetector Detector policies to start the DemoSSHService or the DemoHTTPService services if SSH or HTTP traffic is found in a connection, and drops other connections.

```
Rule(rule_id=1,
    proto=6,
    detect={'DemoSSHDetector' : 'demo_instance/DemoSSHService',
            'DemoHTTPDetector' : 'demo_instance/DemoHTTPService',}
    )
```

6.7.3. Encryption policies

- For a basic overview about Encryption policies, see *Section 6.7.3.1, Understanding Encryption policies (p. 164)*.
- For example, on configuring Encryption policies, see *How to configure SSL proxying in Zorp 7*. For details on HTTPS-specific problems and its solutions, see *How to configure HTTPS proxying in Zorp 7*.
- For details on how the SSL/TLS framework works in Zorp, see *Chapter 3, The Zorp SSL framework* in *Zorp Professional 7 Reference Guide*

6.7.3.1. Understanding Encryption policies

This section describes the configuration blocks of Encryption policies and objects used in Encryption policies. Encryption policies were designed to be flexible, and make encryption settings easy to reuse in different services.

An **Encryption policy** is an object that has a unique name, and references a fully-configured **encryption scenario**.

Encryption scenarios are actually Python classes that describe how encryption is used in a particular connection, for example, both the server-side and the client-side connection is encrypted, or the connection uses a one-sided SSL connection, and so on. Encryption scenarios also reference other classes that contain the actual settings for the scenario. Depending on the scenario, the following classes can be set for the client-side, the server-side, or both.

- **Certificate generator**: It creates or loads an X.509 certificate that Zorp shows to the peer. The certificate can be a simple certificate (*Section 5.5.24, Class StaticCertificate* in *Zorp Professional 7 Reference Guide*), a dynamically generated certificate (for example, used in a keybridging scenario, *Section 5.5.12, Class DynamicCertificate* in *Zorp Professional 7 Reference Guide*), or a list of certificates to support Server Name Indication (SNI, *Section 5.5.18, Class SNIBasedCertificate* in *Zorp Professional 7 Reference Guide*).
  The related parameters are: *`client_certificate_generator`*, *`server_certificate_generator`*
- **Certificate verifier**: The settings in this class determine if Zorp requests a certificate of the peer and the way to verify it. Zorp has separate built-in classes for the client-side and the server-side verification settings: *Section 5.5.6, Class ClientCertificateVerifier* in *Zorp Professional 7 Reference Guide* and *Section 5.5.20, Class ServerCertificateVerifier* in *Zorp Professional 7 Reference Guide*. For details and examples, see *Section 3.2.5, Certificate verification options* in *Zorp Professional 7 Reference Guide*.
  The related parameters are: *`client_verify`*, *`server_verify`*

- **Protocol settings**: The settings in this class determine the protocol-level settings of the SSL/TLS connection, for example, the permitted ciphers and protocol versions, session-reuse settings, and so on. Zorp has separate built-in classes for the client-side and the server-side SSL/TLS settings: *Section 5.5.10, Class ClientSSLOptions* in *Zorp Professional 7 Reference Guide* and *Section 5.5.23, Class ServerSSLOptions* in *Zorp Professional 7 Reference Guide*. For details and examples, see *Section 3.2.6, Protocol-level TLS settings* in *Zorp Professional 7 Reference Guide*.
  The related parameters are: *`client_ssl_options`*, *`server_ssl_option`*

Zorp provides the following built-in encryption scenarios:

- **TwoSidedEncryption**: Both the client-Zorp and the Zorp-server connections are encrypted. For details, see *Section 5.5.25, Class TwoSidedEncryption* in *Zorp Professional 7 Reference Guide*.
- **ClientOnlyEncryption**: Only the client-Zorp connection is encrypted, the Zorp-server connection is not. For details, see *Section 5.5.8, Class ClientOnlyEncryption* in *Zorp Professional 7 Reference Guide*.
- **ServerOnlyEncryption**: Only the Zorp-server connection is encrypted, the client-Zorp connection is not. For details, see *Section 5.5.22, Class ServerOnlyEncryption* in *Zorp Professional 7 Reference Guide*.
- **ForwardStartTLSEncryption**: The client can optionally request STARTTLS encryption. For details, see *Section 5.5.16, Class ForwardStartTLSEncryption* in *Zorp Professional 7 Reference Guide*.
- **ClientOnlyStartTLSEncryption**: The client can optionally request STARTTLS encryption, but the server-side connection is always unencrypted. For details, see *Section 5.5.9, Class ClientOnlyStartTLSEncryption* in *Zorp Professional 7 Reference Guide*.
- **FakeStartTLSEncryption**: The client can optionally request STARTTLS encryption, but the server-side connection is always encrypted. For details, see *Section 5.5.15, Class FakeStartTLSEncryption* in *Zorp Professional 7 Reference Guide*.

For example, on configuring Encryption policies, see *How to configure SSL proxying in Zorp 7*. For details on HTTPS-specific problems and the related solutions, see *How to configure HTTPS proxying in Zorp 7*.

6.7.4. Matcher policies

In general, matcher policies can be used to find out if a parameter is included in a list (or which elements of a list correspond to a certain parameter), and influence the behavior of the proxy class based on the results. Matchers can be used for a wide range of tasks, for example, to determine if the particular IP address or URL that a client is trying to access is on a black or whitelist, or to verify that a particular e-mail address is valid. The matchers usable in a proxy class are described in the *Zorp Professional 7 Reference Guide*.

Note
Matchers can also be used in custom proxy classes created with the **Class editor**.

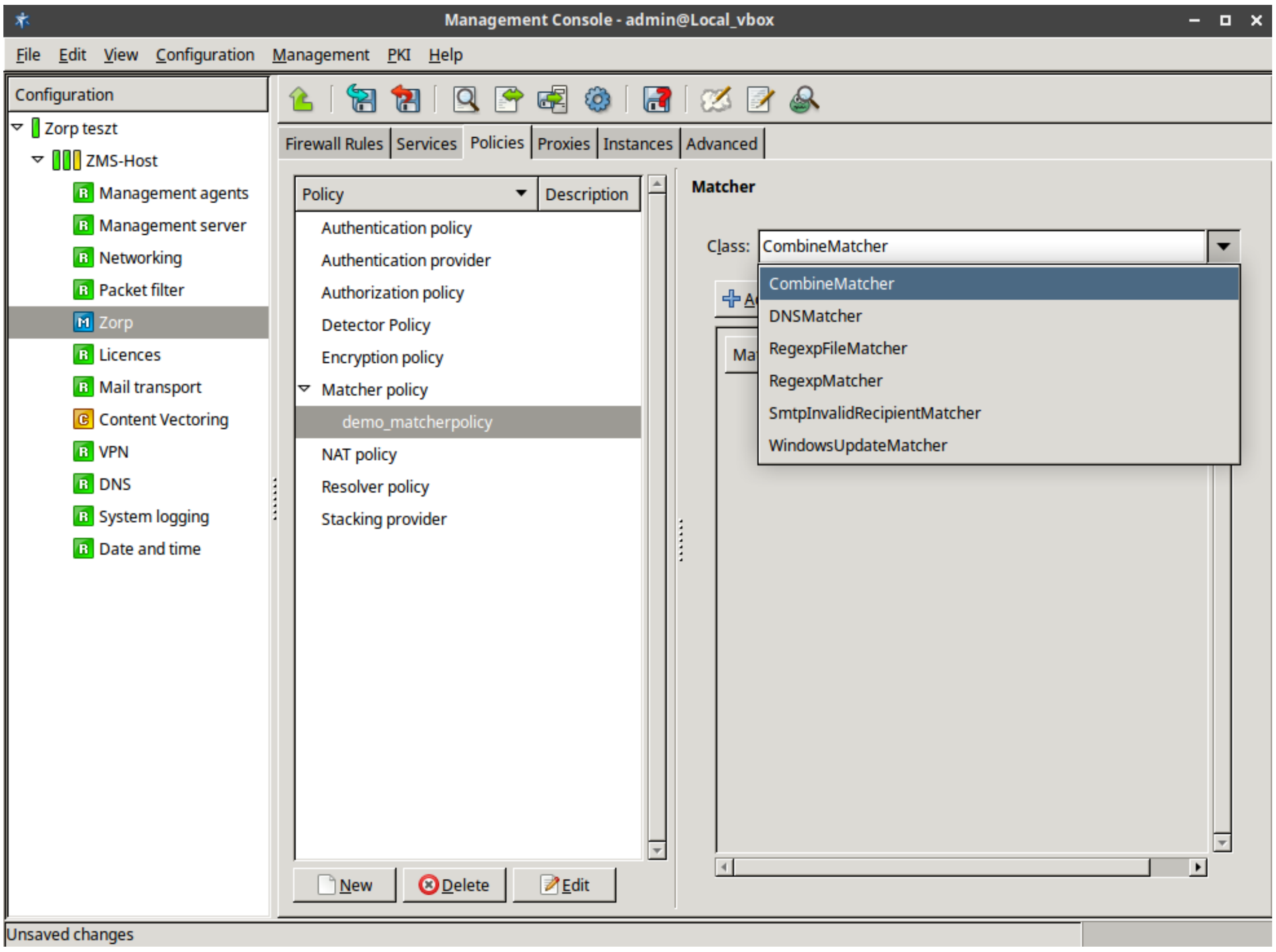


Figure 6.65. Matcher policies

Zorp has a number of predefined matcher classes; and it is also possible to make complex decisions from the results of individual matchers using the *CombineMatcher* class. The available predefined classes are listed below.

- *DNSMatcher*: It retrieves the IP address(es) of a domain from the name server specified.
- *WindowsUpdateMatcher*: It retrieves the IP addresses required for updating computers running Microsoft Windows from the name server specified.
- *RegexpMatcher*: It is a general regular expression matcher.
- *RegexpFileMatcher*: It completes a regular expression based matching on the content of the files.
- *SmtpInvalidRecipientMatcher*: It consults a mail server to verify that an e-mail address is valid.
- *CombineMatcher*: It makes complex decisions by combining the results of multiple simple matchers using logical operations.

Apart from the predefined ones, it is also possible to create custom matcher classes. The various matcher classes and their uses are described in the subsequent sections. The use of matchers in proxy classes is discussed in *Section 6.7.4.7, Using matcher classes in proxy classes (p. 172)*.

6.7.4.1. Matching domain names with DNSMatcher

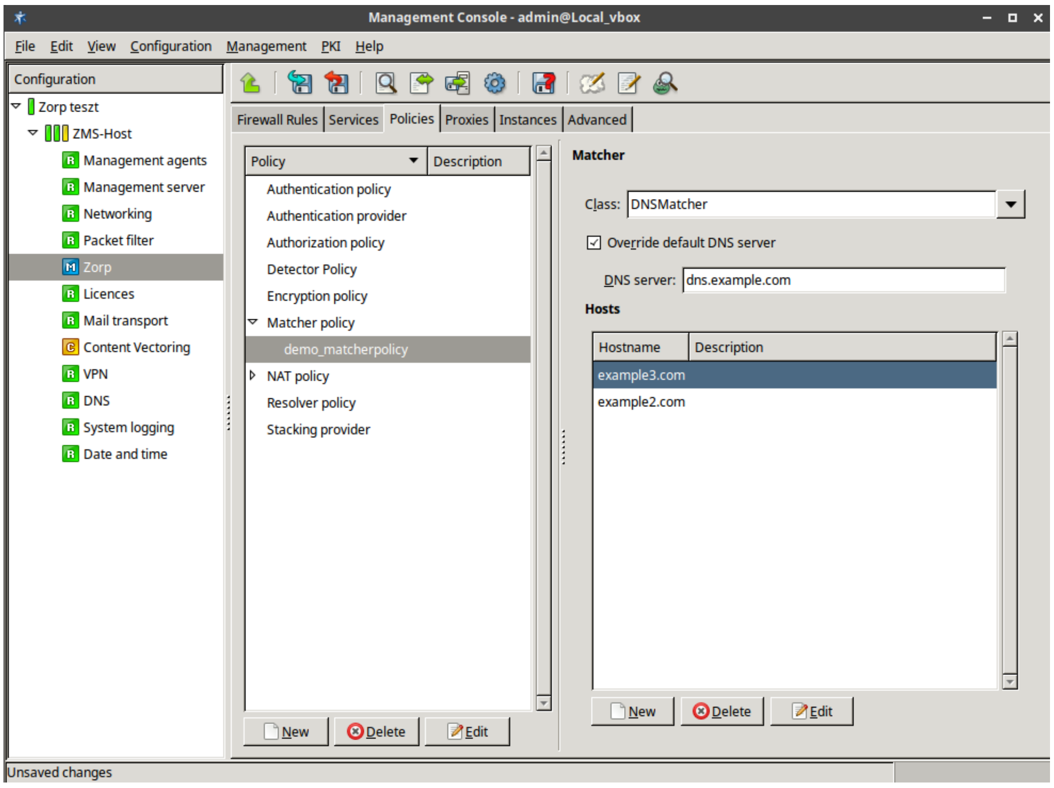
DNSMatcher retrieves the IP addresses of domain names. This can be used in domain name based policy decisions, for example to allow encrypted connections only to trusted e-banking sites. If the IP address of the name server is not specified in the **DNS Server** field, the name server set in the **Networking** component is used (see *Section 5.3, Managing client-side name resolution (p. 81)* for details).

Domain name resolution is completed on-demand basis at each Zorp startup by default, so that unnecessary slowdown with the startup can be avoided. In order to have domain name resolution at each startup, the *`resolve_on_init`* parameter has to be checked in.

Note
Note that in case the zones or the matchers contain unresolvable elements, it may increase the waiting time for a timeout.

It is recommended to have a locally installed caching DNS service which is capable of providing fast responses, monitored with the used domains.

Example 6.10. DNSMatcher for two domain names



Matcher

Class: DNSMatcher

☑ Override default DNS server

DNS server: dns.example.com

Hosts

Hostname	Description
example3.com	
example2.com	

Figure 6.66. Sample DNSMatcher policy

```
Python:
MatcherPolicy(name="ExampleDomainMatcher", matcher=DNSMatcher(server="dns.example.com",\
hosts=("example2.com", "example3.com")))
```

6.7.4.2. WindowsUpdateMatcher

WindowsUpdateMatcher is actually a DNSMatcher used to retrieve the IP addresses currently associated with the *`v5.windowsupdate.microsoft.nsatc.net`*, *`v4.windowsupdate.microsoft.nsatc.net`*, and *`update.microsoft.nsatc.net`* domain names; only the IP address of the name server has to be specified. Windows Update is running on a distributed server farm, using the DNS round robin method and a short TTL to constantly change the set of servers currently visible, consequently the IP addresses of the servers are constantly changing.

Tip
This matcher class is useful to create firewall policies related to updating Windows-based machines. Windows Update is running over HTTPS. For example, there is no real use in inspecting the HTTP traffic embedded into the SSL tunnel (since it is mostly file download), but it is important to verify the identity of the servers.

6.7.4.3. RegexpMatcher

A RegexpMatcher consists of two string lists, one describing the regular expressions to be found (**Match** list) and, optionally, another list of expressions that should be ignored (**Ignore** list) when processing the input. By default, matches are case insensitive. For case sensitive matches, uncheck the **Ignore case** option.

Note
The string lists are stored in the `policy.py` configuration file.

Example 6.11. Defining a RegexpMatcher

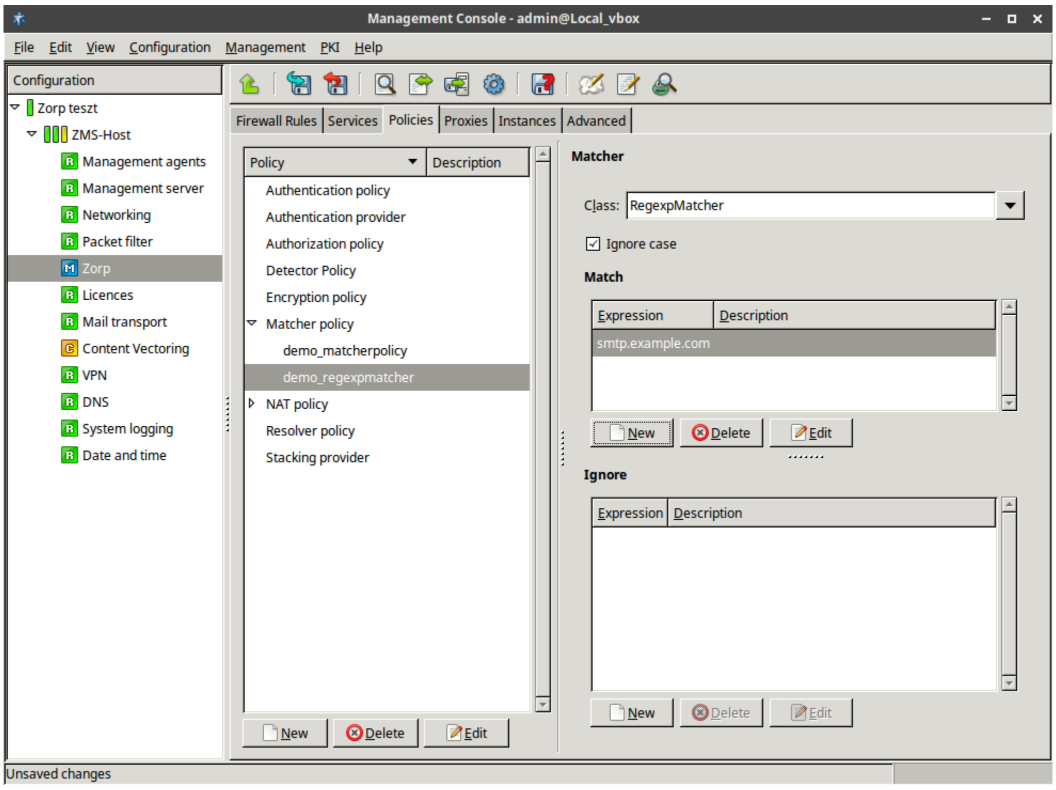


Figure 6.67. Sample RegexpMatcher

The matcher below defines a RegexpMatcher called *`Smtpdomains`*, with only the *`smtp.example.com`* domain in its match list.

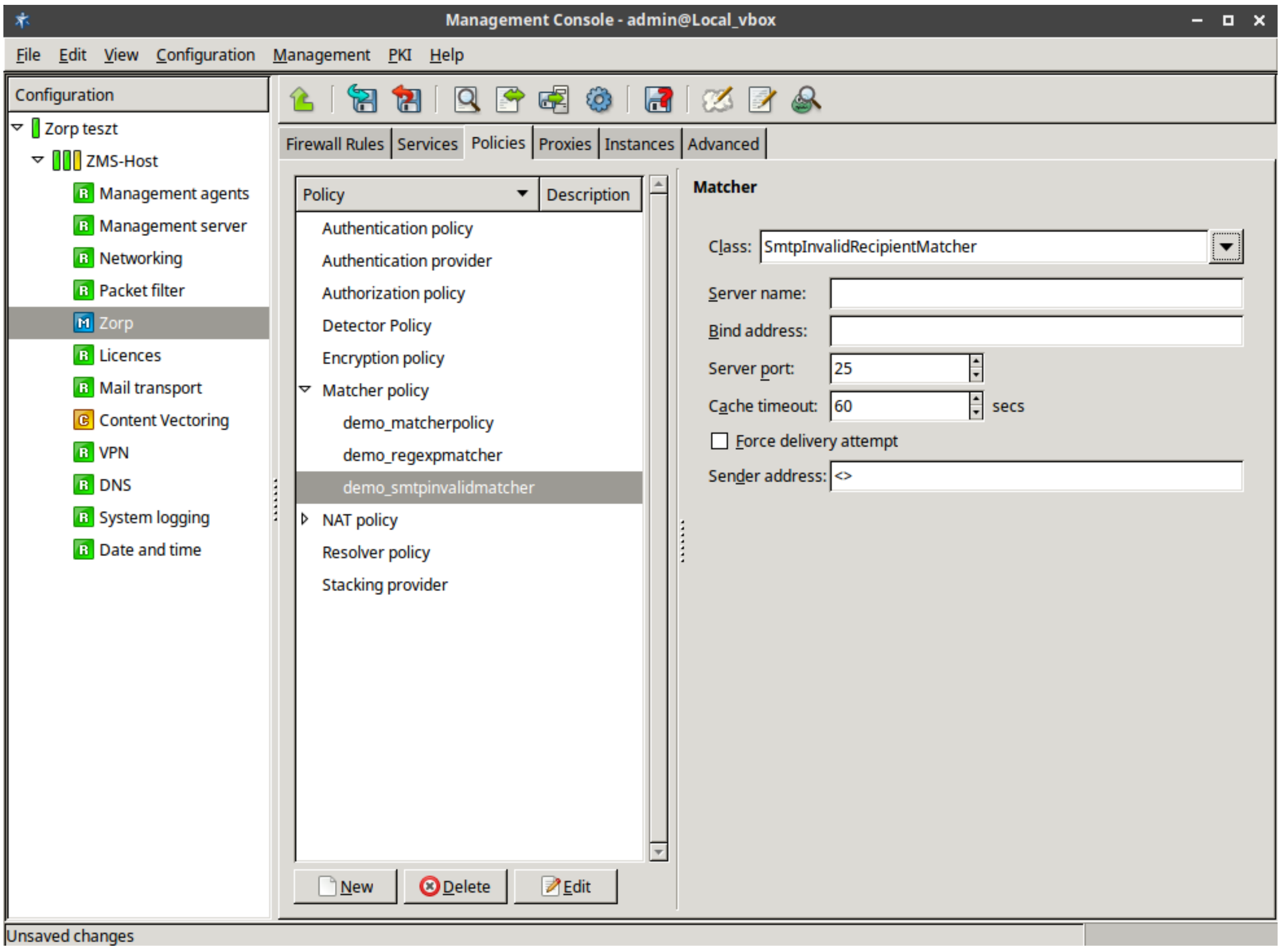
```
Python:
MatcherPolicy(name="Smtpdomains", matcher=RegexpMatcher\
(match_list=("smtp.example.com",), ignore_list=None))
```

6.7.4.4. RegexpFileMatcher

RegexpFileMatcher is similar to RegexpMatcher, the two lists are though stored in separate files. The matcher itself stores only the paths and the filenames to the lists. The files themselves can be managed either manually, or by using the ***FreeText*** plugin.

6.7.4.5. Verifying e-mail addresses with the SmtpInvalidMatcher

This matcher class uses an external SMTP server to verify that a given address (that is, the recipient of an e-mail) exists. The following parameters have to be specified:


Management Console - admin@Local_vbox

Matcher

Class: SmtpInvalidRecipientMatcher

Server name:

Bind address:

Server port: 25

Cache timeout: 60 secs

☐ Force delivery attempt

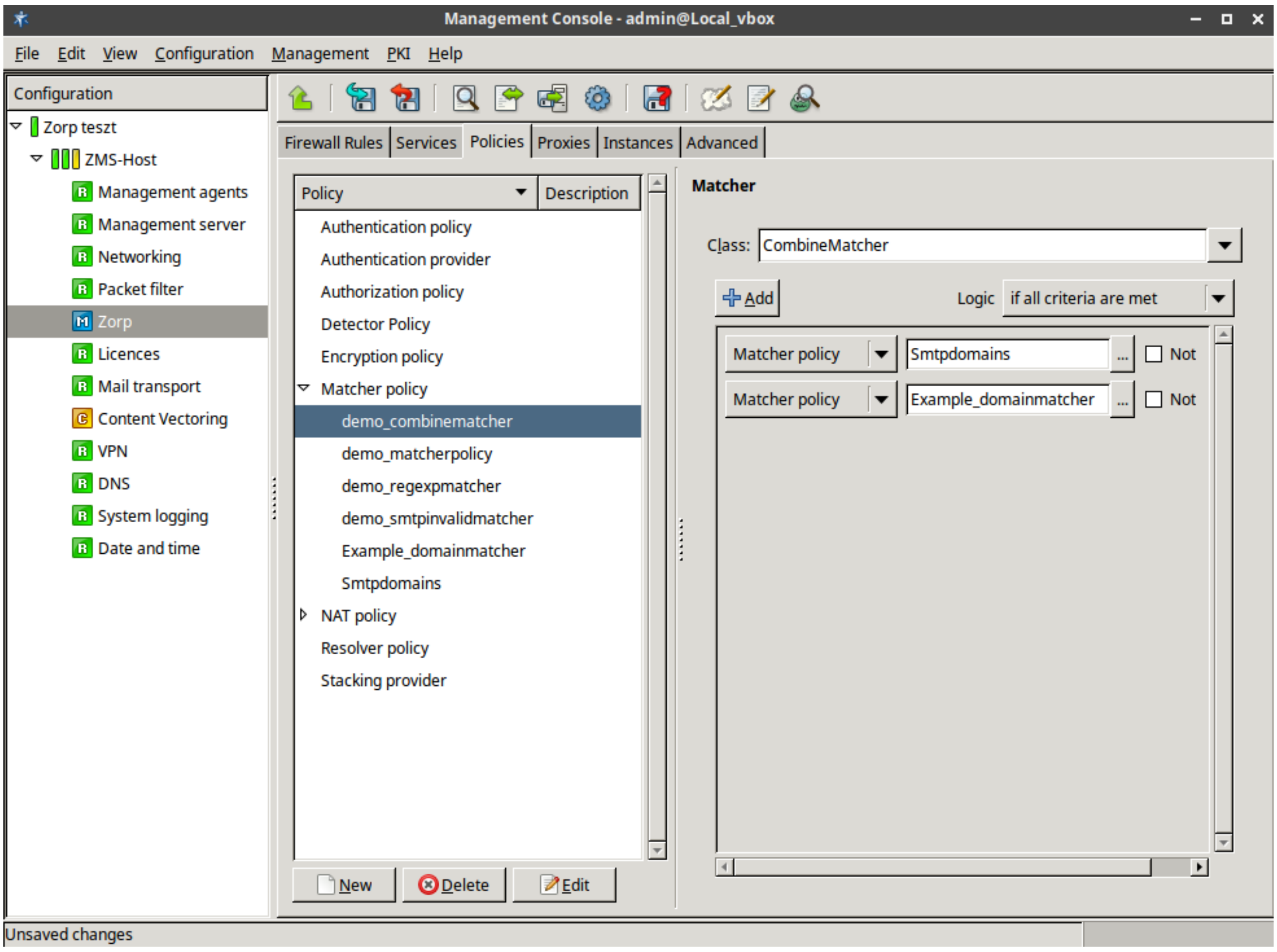
Sender address: <>

Figure 6.68. Configuring SmtpInvalidMatcher

- **Server name** and **Server port**: These fields identify the domain name and the port used by the SMTP server to be queried.
- **Bind address**: Zorp uses this IP address as the source address of the connection when connecting the SMTP server to be queried.
- **Cache timeout**: The result of a query is stored for the period set as **Cache timeout** in seconds. If a new e-mail is sent to the same recipient within this interval, the stored verification result is used.
- **Force delivery attempt** and **Sender address**: Send an e-mail to the recipient as if it were sent by **Sender address**. If the destination mail server accepts this mail, the original e-mail is sent. This option is required because many mail server implementations do not support the `VRFY` SMTP command properly, or it is not enabled. (The default value for **Sender address** is <>, which refers to the mailer daemon and is recognized by virtually all servers.)

6.7.4.6. Making complex decisions with the CombineMatcher

CombineMatcher uses the results of multiple matchers to make a decision. The results of the individual matchers can be combined using the logic operations AND, OR and NOT. Both existing matcher policies (**Matcher policy**) and policies defined only within the particular CombineMatcher (**Matcher instance**) can be used.



Matcher

Class: CombineMatcher

Add

Logic: if all criteria are met

Matcher policy	Smtpdomains	☐ Not
Matcher policy	Example_domainmatcher	☐ Not

Figure 6.69. CombineMatcher

New matchers can be added with the **Add** button. Each line in the main window of the CombineMatcher configuration panel corresponds to a matcher. To use an existing matcher policy, set the combobox on the left to **Matcher policy**. After that, clicking the **...** icon opens a dialog window where the matcher to be used can be selected.

Tip
Matchers can also be configured whenever needed: set the combobox to **Matcher instance**, click on the **...** icon, and select and configure the matcher as required.

Each matcher can be taken into consideration either with its regular, or with its inverted result using the **Not** checkbox. The individual matchers (or their inverses) can be combined with the logical AND, and OR operations. This can be set in the **Logic** combobox:

- **If all criteria are met**: It corresponds to the logical AND; the CombineMatcher will return the TRUE value only if all criteria are TRUE.

- **If any criteria are met**: It corresponds to the logical OR; the CombineMatcher will return the TRUE value if at least one criteria is TRUE.
- **If all criteria are the same**: CombineMatcher will return the TRUE value if all criteria have the same value (either TRUE or FALSE).

Tip
A CombineMatcher can also be used to combine the results of other CombineMatchers, thus *very* complex decisions can also be made.

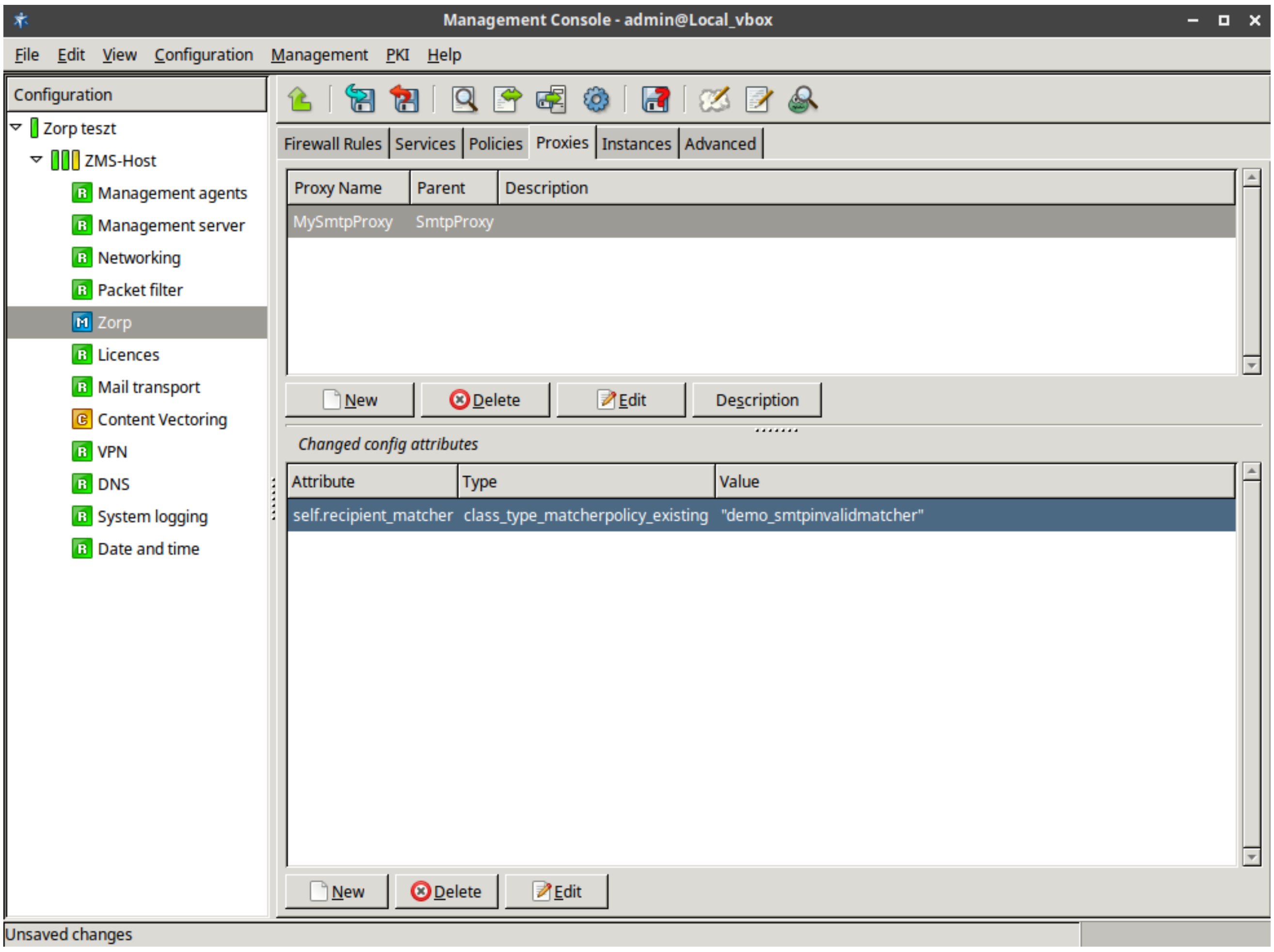
Example 6.12. Blacklisting e-mail recipients
A simple use for CombineMatcher is to filter the recipients of e-mail addresses using the following process:

1. An SmtpInvalidMatcher (called *`SmtpCheckrecipient`*) verifies that the recipient exists.
2. A RegexpMatcher (called *`SmtpWhitelist`*) or RegexpFileMatcher is used to check if the address is on a predefined list. This list is either a whitelist (permitted addresses) or a blacklist (addresses to be rejected).
3. A CombineMatcher (called *`SmtpCombineMatcher`*) sums up the results of the matchers with a logical AND operation (**If all criteria are met**). If the list of the RegexpMatcher is a blacklist, the result of the RegexpMatcher should be inverted by checking the **Not** checkbox.
4. An SmtpProxy (called *`SmtpRecipientMatcherProxy`*) references *`SmtpCombineMatcher`* in its *`recipient_matcher`* attribute.

```
Python:
class SmtpRecipientMatcherProxy(SmtpProxy):
  recipient_matcher="SmtpCombineMatcher"
  def config(self):
    SmtpProxy.config(self)

MatcherPolicy(name="SmtpCombineMatcher", matcher=CombineMatcher\
(expr=(Z_AND, "SmtpCheckrecipient", "SmtpWhitelist")))
MatcherPolicy(name="SmtpWhitelist", matcher=RegexpMatcher\
(match_list=("info@example.com",), ignore_list=None))
MatcherPolicy(name="SmtpCheckrecipient", matcher=SmtpInvalidRecipientMatcher\
(server_port=25, cache_timeout=60, attempt_delivery=FALSE, \
force_delivery_attempt=FALSE, server_name="recipientcheck.example.com"))
```

6.7.4.7. Using matcher classes in proxy classes

To actually use the defined matchers, they have to be specified in the proper attributes of the particular proxy class. The proxy classes can use matchers for a variety of tasks. The matchers to be used by the particular proxy class can be added as **Changed class attributes**.


Management Console - admin@Local_vbox
File Edit View Configuration Management PKI Help
Configuration
Zorp teszt
ZMS-Host
Management agents
Management server
Networking
Packet filter
Zorp
Licences
Mail transport
Content Vectoring
VPN
DNS
System logging
Date and time
Firewall Rules Services Policies Proxies Instances Advanced
New Delete Edit Description
Changed config attributes
New Delete Edit
Unsaved changes

Proxy Name	Parent	Description
MySmtpProxy	SmtpProxy	

Attribute	Type	Value
self.recipient_matcher	class_type_matcherpolicy_existing	"demo_smtpinvalidmatcher"

Figure 6.70. Using matchers in proxy classes

For example, SmtpProxy can use three different matchers: one to check the relayed domains (for example, using a *RegexpMatcher*), and two *SmptInvalidMatchers* (one for sender, one for the recipient address verification). For the details of the matchers usable by a given proxy class see the attributes of the class in *Chapter 4, Proxies* in *Zorp Professional 7 Reference Guide*.

Example 6.13. SmtpProxy class using a matcher for controlling relayed zones

```
Python:
class SmtpMatcherProxy(SmtpProxy):
  relay_domains_matcher="Smtpdomains"
  def config(self):
    SmtpProxy.config(self)

MatcherPolicy(name="Smtpdomains", matcher=RegexpMatcher(match_list=("example.com",),\
 ignore_list=None))
```

6.7.5. NAT policies

Network Address Translation (NAT) is a technology that can be used to change source or destination addresses in a connection from one IP address to another one.

Today, most corporate networks work with private, non-Internet-routable IP addresses from the `10.0.0.0/8`, `172.16.0.0/12` or `192.168.0.0/16` ranges. These IP addresses are suitable for intranet communication (they can even be routed internally) but cannot be used on the Internet. Therefore a mechanism, NAT, is in place, that, whenever an internal client wants to access Internet resources, uses a public IP address for this purpose.

Another use for NAT is address hiding. Towards the Internet only a limited number of public IP addresses are visible, while you may have thousands of internal clients and servers all “translated” to those few addresses.

Tip
Address hiding with the help of NAT is especially useful for published servers (usually located in a DMZ). The potential attackers can see only a few public IP addresses (or even a single one) but from this information the actual location and the number of the servers, or even their configuration details remain hidden.

Note
Although address hiding can be considered as a security feature, NAT alone is not enough to protect a network from malicious intruders. Never use NAT in itself as a security solution.

Based on whether source or destination IP addresses are transformed, two kinds of NAT can be differentiated.

- Source NAT (SNAT) - if source IP addresses are transformed
- Destination NAT (DNAT) - if address transformation is performed on destination IP addresses

Basically, NAT in Zorp provides a possibility to modify the source and destination IP address at the server side before the connection is established. The source and destination IP address values are defined previously by the router on the basis of the router type, its settings and the client request. In some cases, Proxy settings can also be involved, for example when the **Target address overrideable by the proxy** router option is enabled. NAT is responsible for changing IP addresses to some other addresses depending on the configuration of the NAT policy.

Note
This NAT is performed by the **Zorp** component and is totally different from the NAT offered by IPTables in the **Packet Filter** component.

By default, when no NAT policy is set up, Zorp uses its own IP address configured for the corresponding network interface when sending out traffic in any direction. Although this results in many clients “using” the same (single) IP address, it is not considered as SNAT technically.

In Zorp, NAT is performed on the application proxy level. In this case it is not strictly IP address replacement, since original packets do not appear in the traffic leaving the firewall. It is an IP address modification rule rather. If application proxying is used — meaning there is no traffic that is forwarded on the packet filter level — packet filter level NAT is neither required nor recommended.

Service definitions contain NAT policy settings to configure application proxy-level NAT.

Note
If some traffic is managed at the packet filter level, NAT must be performed at the packet filter level, too. In this case, NAT is actually an IP address replacement, since the packet filter – rules permitting – forwards the original packets it receives. You are not recommended to use NAT at the packet filter level.

For more details on packet filter-level NAT solutions, see *Appendix A, Packet Filtering (p. 462)*.

Originally, NAT meant only address translation, leaving port information travelling in TCP or UDP packets unmodified. This can lead to errors especially in large networks where thousands of clients are NATed by a single machine, therefore NAT technologies were modified to do port translation as well, guaranteeing that even if two internal clients try to use the same source port number in session establishments, the packets leaving the NAT server always have unique source IP address/port pairs.

Note
Port translating NAT devices are incompatible with IPSec encryption (which is a major drawback as IPSec is becoming increasingly popular) and may be incompatible with proprietary, two-channel protocols as well.

6.7.5.1. Configuring NAT in Zorp

Before you start NAT configuration you must decide whether you need it at all. If you need traffic redirection, for example a Web server in your DMZ, routers may serve your needs. By default, Zorp uses its own IP address (bound to the corresponding adapter) to all connections leaving it in any direction, unless the **Use client address as source** router option is set, in which case the original client IP address is used. Consequently, NAT may not be absolutely necessary.

Note
Configuring **SNAT Policy** for a **Service** automatically enables the **Use client address as source** router function, so during SNAT the client's address is used, not the firewall's.

As opposed to network configurations without firewalls, where NAT is a universal setting for all clients communicating with any protocol, in Zorp, different traffic can be NATed differently because NAT configurations are linked to services. It can happen that while outgoing HTTP traffic is SNATed to a single public IP address, SQL traffic from the same network is not SNATed at all, and finally FTP download traffic is SNATed to a separate NAT pool.

6.7.5.1.1. Procedure – Configuring NAT

Step 1. Create the required NAT policies on the **Policies** tab of the **Zorp** ZMC component.

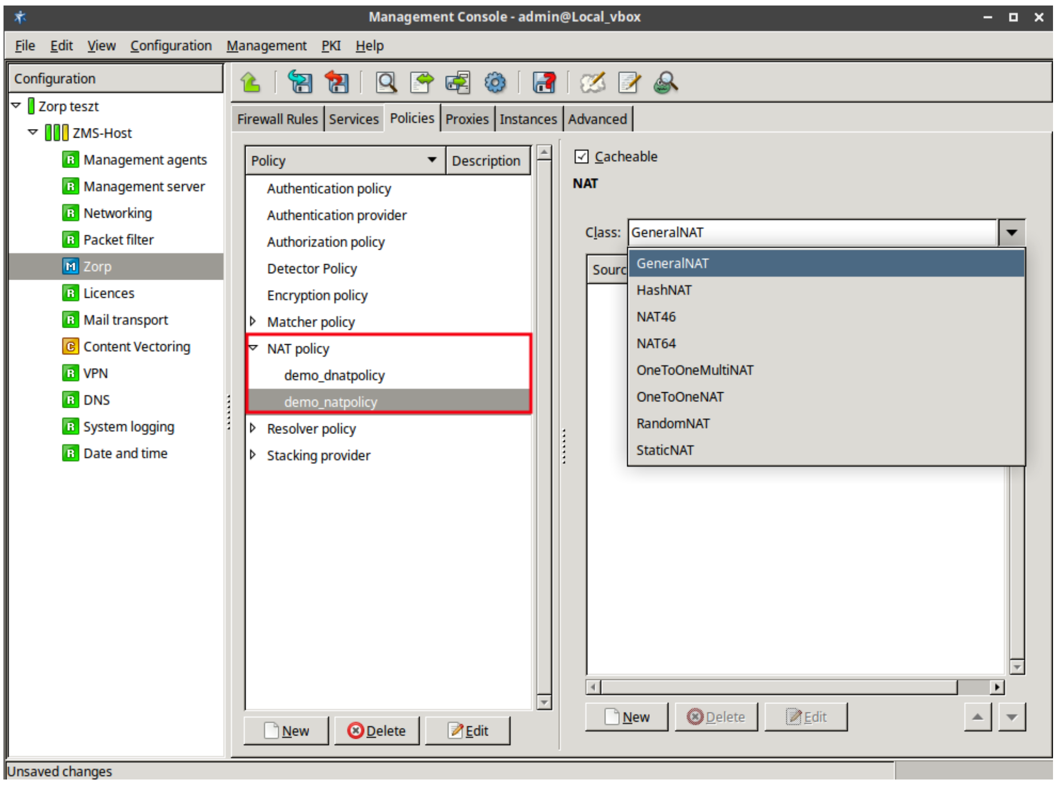


Figure 6.71. NAT policy configuration window

Click the **New** button, select **NAT Policy** from the **Policy type** combobox and supply a name for the new policy.

Names should be descriptive and ideally contain information about the direction of NATing and/or about the type of traffic NATed.

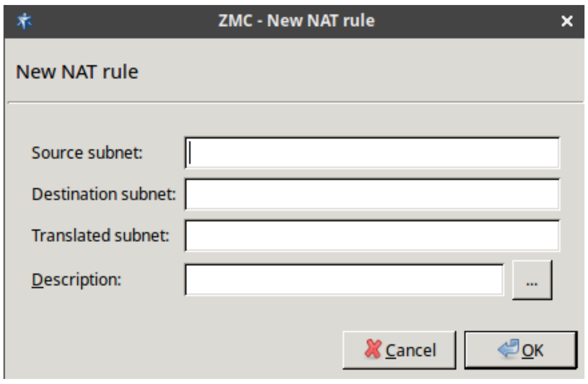
In most network configurations NAT is typically not service-specific; a generic NAT policy may be adequate for most outgoing or incoming traffic. There are no compulsory rules for naming.

NAT policies can be renamed any time.

Tip
Remove NAT policies from the configuration set if they are no longer needed.

NAT policies can be removed only if they are not used in any service definition.

Step 2. Configure a NAT solution.
Zorp supports several different NAT solutions.



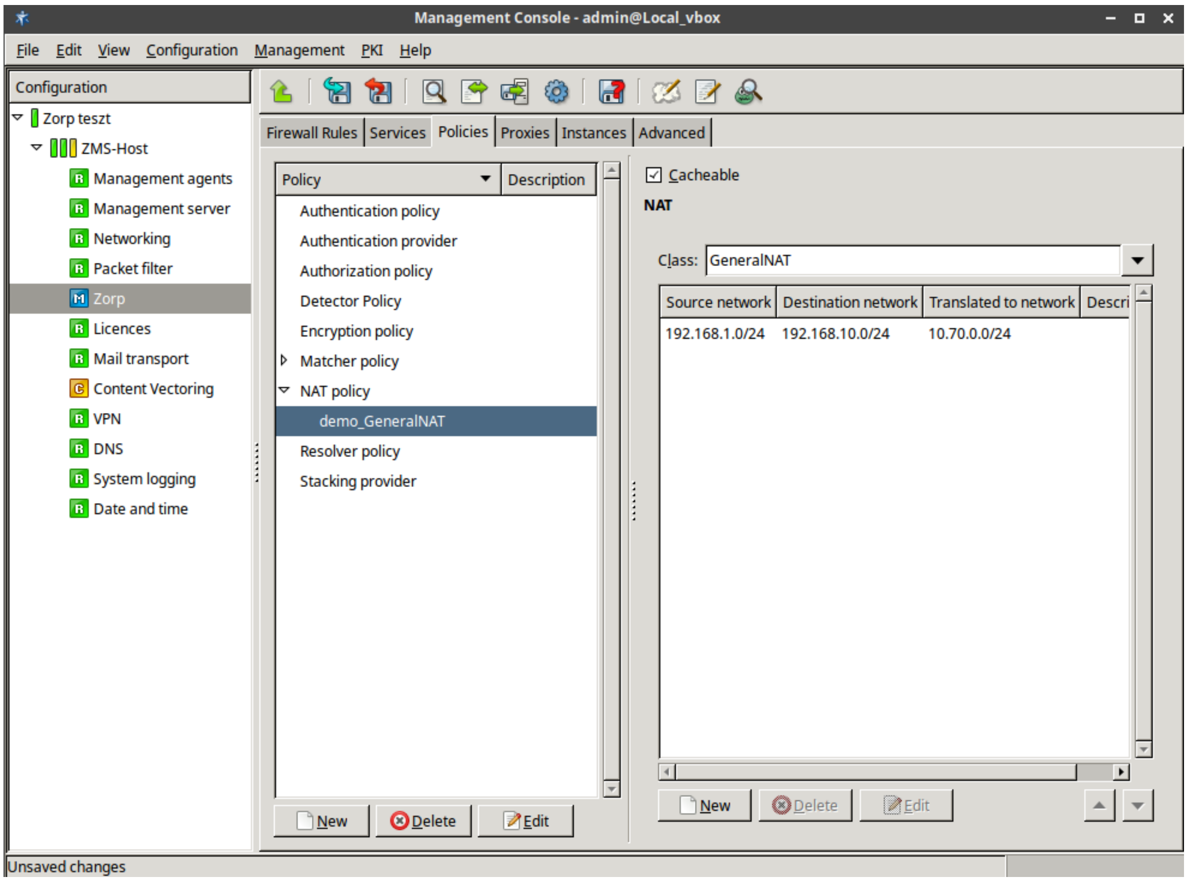
ZMC - New NAT rule

New NAT rule

Source subnet:	
Destination subnet:	
Translated subnet:	
Description:	

Figure 6.72. Editing a GeneralNAT rule

GeneralNAT has three parameters: source subnet, destination subnet, and the translated subnet. Connections arriving from the source subnet, that target the destination subnet, are modified to use the translated subnet. If the NAT policy is used as Source-NAT (SNAT), the source subnet is translated to translated subnet, if the policy is used as Destination NAT (DNAT), the target subnet is translated.



Management Console - admin@Local_vbox

☑ Cacheable

NAT

Class: GeneralNAT

Source network	Destination network	Translated to network	Descri
192.168.1.0/24	192.168.10.0/24	10.70.0.0/24	

Figure 6.73. Sample GeneralNAT rule

Note
The original and translated netmasks do not need to be the same: it is possible to map an entire /24 network onto a single IP address (/32 mask). However, the order of the pairs is important because the Zorp processes the list from top to bottom.

Depending on the NAT type (SNAT, DNAT), Zorp evaluates the NAT rules one after the other. If a row containing the address to be NATed as the source network, the iteration stops and Zorp modifies the IP address as specified in that row. If no match is found, the original IP address is used.

When modifying the address, it calculates the host ID of the address using the target netmask and the source network address and adds it to the target network.

Example 6.14. Address translation examples using GeneralNAT
The following two tables show a number of simple and special **GeneralNAT** cases. The Destination Address in these cases is set to *`0.0.0.0/0`*.

Source network	Target network	Source IP address	Translated IP address
`10.0.1.0/24`	`192.168.1.0/24`	`10.0.1.5`	`192.168.1.5`
`10.0.1.0/24`	`192.168.1.0/25`	`10.0.1.130`	`192.168.1.2`
`10.0.1.0/25`	`192.168.1.0/24`	`10.0.1.42`	`192.168.1.42`

Source network	Target network	Original IP address	New IP address
`0.0.0.0/0`	`192.168.2.2/32`	`172.17.3.5`	`192.168.2.2`
`0.0.0.0/0`	`192.168.3.0/31`	`172.18.1.1`	`192.168.3.1`
`0.0.0.0/0`	`192.168.3.0/31`	`172.18.1.2`	`192.168.3.0`
`192.168.3.1/32`	`172.19.2.0/24`	`192.168.3.1`	`172.19.2.0`

Step 3. Configure caching.
Since the NAT decision may take a long time in some cases (for example, if there are many mappings in the list), the decisions can be stored in a cache. Storing the decisons in a cache accelerates future decisions. Caching can be enabled/disabled using the **Cacheable** checkbox in the configuration window of the NAT policy.

Tip
It is recommended to enable caching for complex NAT decisions.

Note
Enabling caching can have interesting, but sometimes unwanted effects on some NAT types, for example on `RandomNAT`. Using `RandomNAT` and the `Cachable` option together would result in Load Balancing, but with sticky IP addresses, the cache would remember which source IP address was used before for a specific client's IP address and would use the same address again. This can be very useful, but can also cause a lot of problems and troubleshooting.

6.7.5.2. Types of NAT policies

Zorp supports the following types of NAT policies. For details on the parameters of these NAT policies, see *Section 5.8, Module NAT* in *Zorp Professional 7 Reference Guide*.

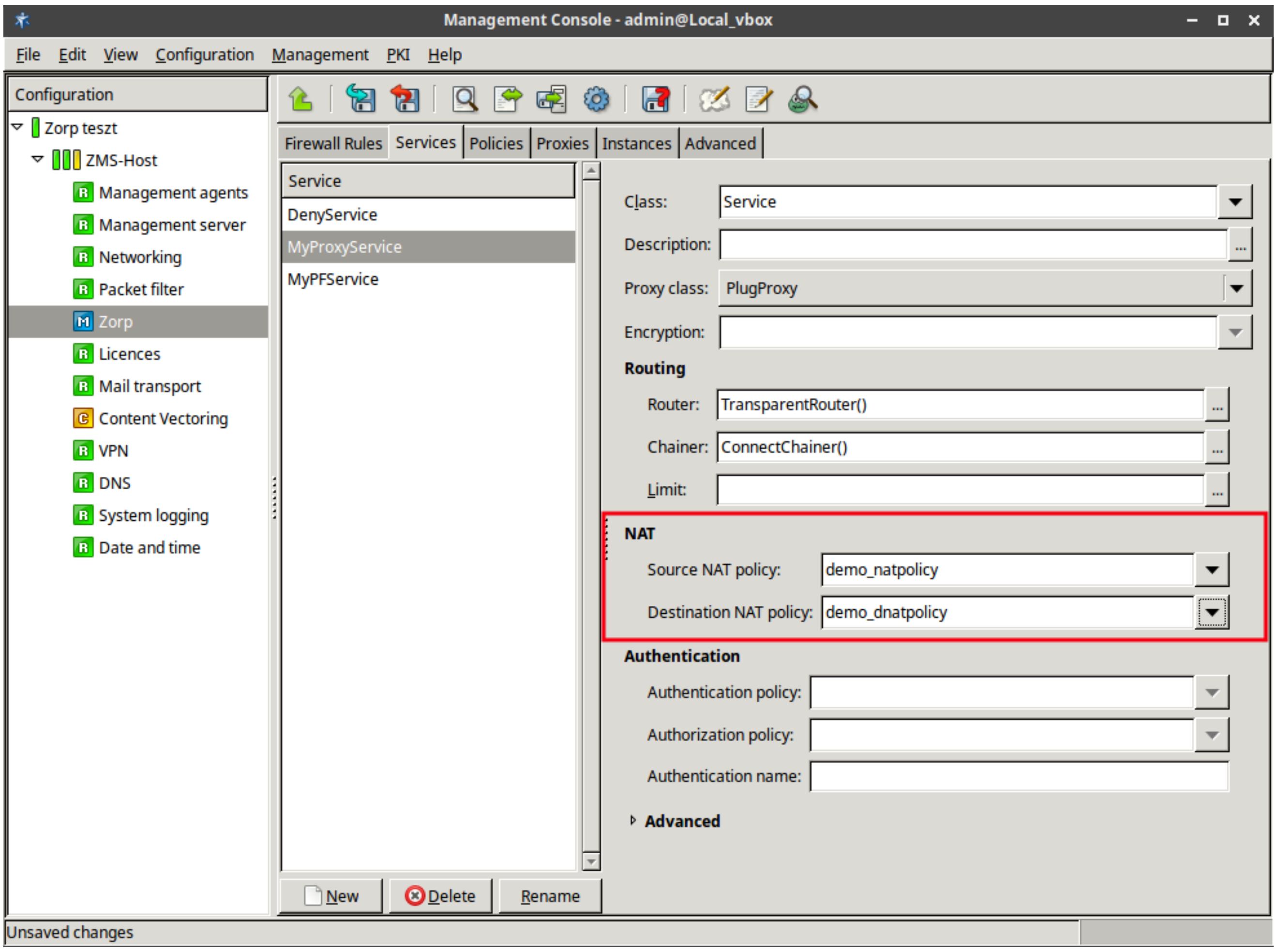
NAT policy	Description
GeneralNAT	This options means a simple mapping based on the original and desired address(es). GeneralNAT can be used to map a set of IP addresses (a subnet) to either a single IP address or to a set of IP addresses (a subnet). For details, see *Section 5.8.3, Class GeneralNAT* in *Zorp Professional 7 Reference Guide*.
StaticNAT	This option can be used to specify a single IP address/port pair to be used in address transforms. It is mainly used in DNAT configurations where incoming traffic must be directed to an internal or DMZ server that has a private IP address. Specifying port translation is optional. When used in conjunction with SNAT, StaticNAT can be used to map to IP alias(es). For details, see *Section 5.8.11, Class StaticNAT* in *Zorp Professional 7 Reference Guide*.
OneToOneNAT	In OneToOneNAT mapping you must configure IP address mappings for your address sets (domains) individually. In other words, OneToOneNAT maps networks to networks — with the possibility that your networks consist of single IP addresses. To use OneToOneNAT the two networks must be of the same size. For details, see *Section 5.8.9, Class OneToOneNAT* in *Zorp Professional 7 Reference Guide*.
OneToOneMultiNAT	This option maps multiple IP address domains to multiple IP address domains. It is primarily useful for large-scale, enterprise deployments. It is like OneToOneNAT but can have multiple NAT mappings. For details, see *Section 5.8.8, Class OneToOneMultiNAT* in *Zorp Professional 7 Reference Guide*.
RandomNAT	In case of this option the firewall selects an IP address from the configured NAT pool randomly for each new connection attempt. Once a communication channel (a session) is established, subsequent packets belonging to the same session use the same IP address. The tranform of the port number used in RandomNAT can be fixed, even for each IP address used in the NAT pool separately. It is ideal when you want to distribute

NAT policy	Description
	the load (use) of addresses in your NAT pool evenly and you do not have specific requirements for fixed address allocations such as IP based authentication. For details, see *Section 5.8.10, Class RandomNAT* in *Zorp Professional 7 Reference Guide*.
HashNAT	It maps individual IP addresses to individual IP addresses very quickly, using hash values to determine mappings and storing them in hash tables. For details, see *Section 5.8.4, Class HashNAT* in *Zorp Professional 7 Reference Guide*.
NAT46	NAT46 embeds an IPv4 address into a specific portion of the IPv6 address, according to the NAT46 specification described in *RFC6052*. For details, see *Section 5.8.5, Class NAT46* in *Zorp Professional 7 Reference Guide*.
NAT64	NAT64 maps specific bits of the IPv6 address to IPv4 addresses according to the NAT64 specification described in *RFC6052*. For details, see *Section 5.8.6, Class NAT64* in *Zorp Professional 7 Reference Guide*.

Table 6.3. NAT solutions

6.7.5.3. NAT and services

The NAT policies created in the **Policies** tab can be used in service definitions. Navigate to the **Services** tab, select a service and choose a NAT policy as either the Source NAT policy or the Destination NAT policy service parameter.



Management Console - admin@Local_vbox

File Edit View Configuration Management PKI Help

Configuration
- Zorp teszt
  - ZMS-Host
    - Management agents
    - Management server
    - Networking
    - Packet filter
    - Zorp
    - Licences
    - Mail transport
    - Content Vectoring
    - VPN
    - DNS
    - System logging
    - Date and time

Firewall Rules | Services | Policies | Proxies | Instances | Advanced

Service
- DenyService
- MyProxyService
- MyPFService

New | Delete | Rename

Class: Service
Description:
Proxy class: PlugProxy
Encryption:

Routing

Router: TransparentRouter()
Chainer: ConnectChainer()
Limit:

NAT

Source NAT policy: demo_natpolicy
Destination NAT policy: demo_dnatpolicy

Authentication

Authentication policy:
Authorization policy:
Authentication name:

Advanced

Unsaved changes

Figure 6.74. Using a NAT policy in a service definition

Remember that NAT policies are independent configuration entities and come into effect only if they are used in service definitions. Also, SNAT and DNAT policies are two different and independent service parameters: it is not required to have either one or both in any service definition. One service can only use a single NAT policy (or none) as its Source and another one (or none) as its Destination NAT policy parameter. These two settings usually do not reference the same NAT policy (although this is not impossible).

In general, while all NAT policies are equal in that they are freely usable as either source or destination NAT policies in service definitions, they are typically created with their future use in mind. There is no specification on whether NAT policies are SNAT or DNAT policies: they are SNAT or DNAT policies only from the point of view of the services that are using them.

NAT policies can be reused. Any number of services can use them.

Note
Although it is often considered a security-enhancing feature, NAT is not intended for access control of any type. Instead, use proper Zone setups and service definitions for this purpose.

6.7.5.4. NAT and other policy objects

NAT in Zorp is an option to change source/destination IP address information in the server side of connections of the firewall, immediately before the connection is started. Since NAT decisions (if used) are made after all other IP address related configurations, such as router, proxy configuration, NAT can override these previous settings. NAT in Zorp can be used to shift IP address ranges, to set IP addresses and to customize these operations.

In a service definition there are potentially two different components that directly deal with IP address setting:

- a router (compulsory),
- and a NAT policy (or two) (optional).

In the address setting procedure the following processes are involved.

1. Incoming connection is accepted and a new session is created.
2. Destination address is set by the Router and using the **Use client address as source** option the source address of the server side connection is also set.
   Remember that the Router only gives a suggestion for the source/destination IP addresses as the proxy or the NAT can later override these suggestions.
3. Router settings can be altered by the proxy if the **Target address overrideable by the proxy** option is set or `InbandRouter` is selected and the proxy has some protocol-based information.
4. NAT is performed, depending on NAT types (SNAT/DNAT).
5. Access control check is performed based on the final destination IP address decision. Check whether the service is allowed as an `inbound service` into the zone where the destination IP address belongs to.
6. The connection to the server is established.

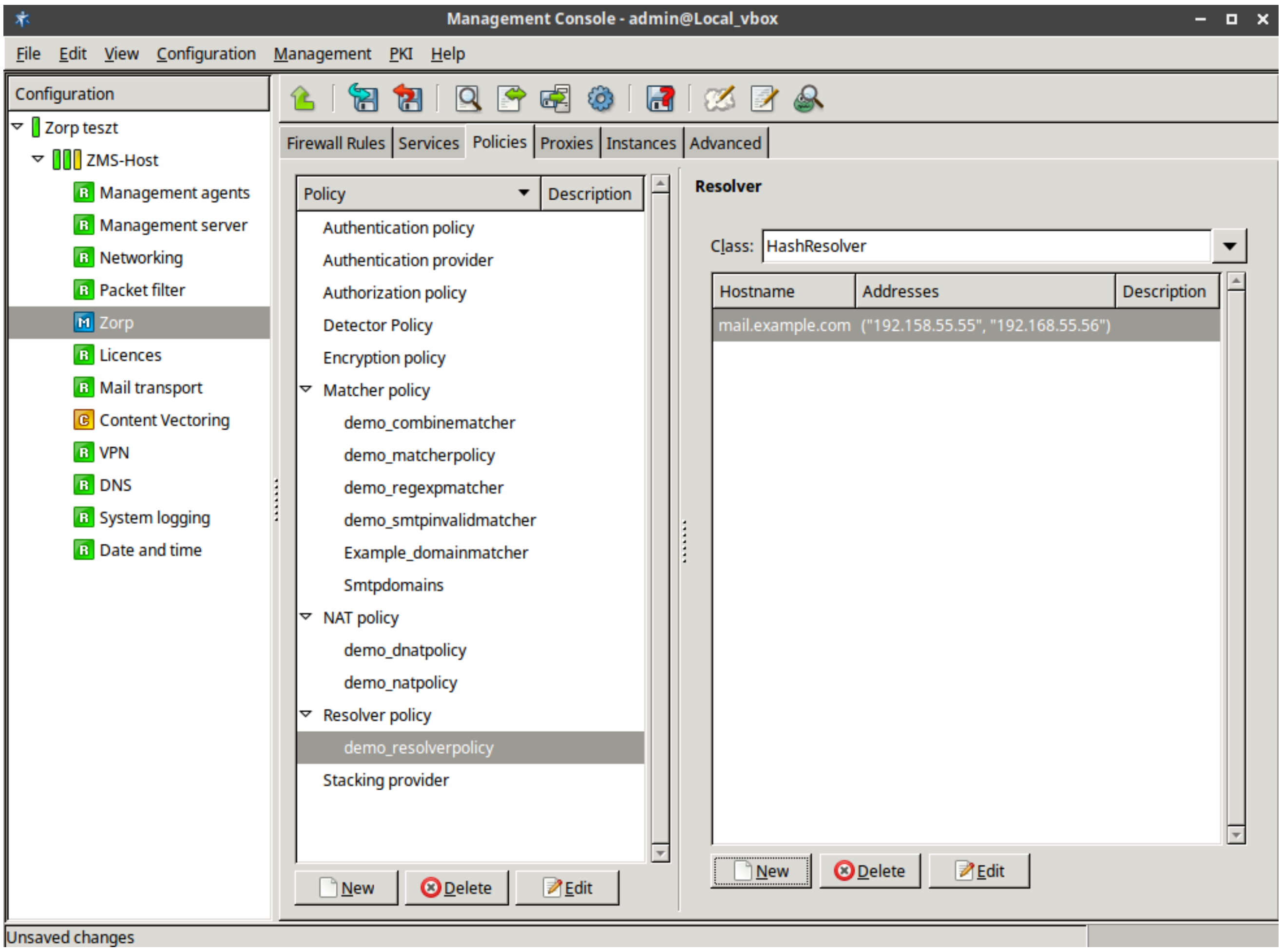
Note
When checking the inbound services of the zone, the IP address to which the firewall actually connects to is considered. In other words, the original destination address of the client may be overridden by the router, the proxy and DNAT as well. Zone access control uses only the final IP address, all interim addresses (set by the Router, Proxy, but not used as the final one) are ignored in the access control decision.

If a service uses an SNAT Policy, the **Use client address as source** is implicitly set as well so that SNAT uses the client IP address instead of the firewall IP address. That is, if the NAT policy does not include SNAT modification, the client's IP address is used even if the **Use client address as source** is unset in the router.

Tip
The versatility of NAT policies is especially useful in large-scale, enterprise deployments or where a lot of NAT is used.

6.7.6. Resolver policies

Resolver policies specify how a given service should resolve the domain names in client requests. This capability is essential when non-transparent services are used, as in these cases the Zorp host has to determine the destination address, and the results of a name resolution are needed. Zorp is also able to store the addresses of often used domain names in a hash. Zorp supports DNS-based (**DNSResolver**) and Hash table-based (**HashResolver**) name resolution.



Class: HashResolver

Hostname	Addresses	Description
mail.example.com	("192.158.55.55", "192.168.55.56")	

Figure 6.75. Resolver policies

DNSResolver policies query the domain name server used by Zorp in general to resolve domain names. If a domain name is associated to multiple IP addresses (that is, it has more than one 'A' records), these records can be retrieved by checking the **Return multiple DNS records** checkbox. From Zorp version 7.0 12, the **DNSResolver** policies also cache the domain names and the IP addresses found. (The DNS server used by the Zorp host can be specified on the **Resolver** tab of the **Networking** component, see *Section 5.3, Managing client-side name resolution (p. 81)* for details.)

Tip
Retrieving multiple 'A' records is useful when Zorp is used to perform load balancing.

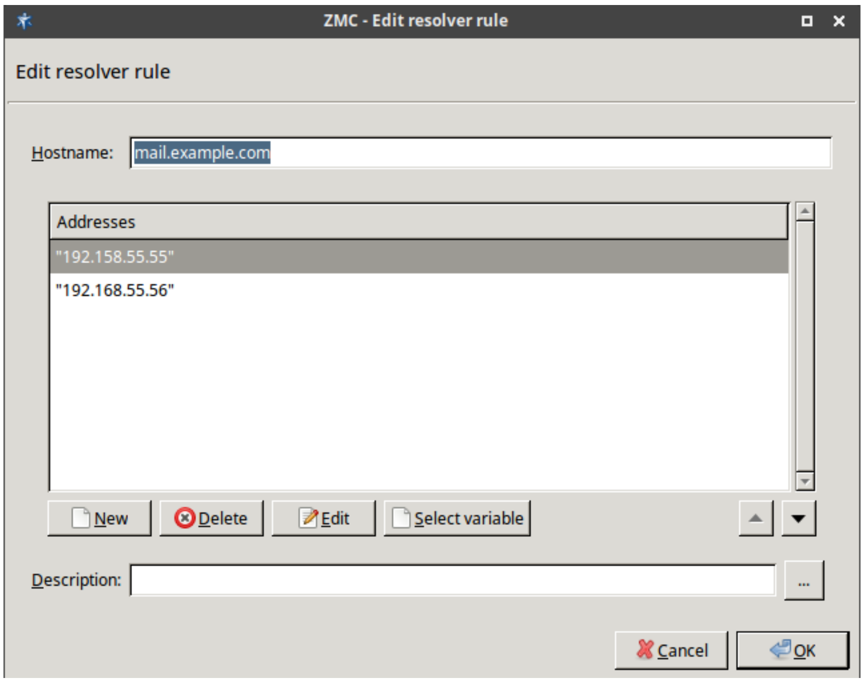
Example 6.15. Defining a Resolver policy
Python: Below is a simple DNSResolver policy enabled to return multiple 'A' records.

```
ResolverPolicy(name="Mailservers", resolver=DNSResolver(multi=TRUE))
```

HashResolver policies are used to locally store the IP addresses belonging to a domain name. A domain name (**Hostname**) and one or more corresponding IP addresses (**Addresses**) can be stored in a hash. If the domain name to be resolved is not included in the hash, the name resolution will fail. The **HashResolver** can be used to direct incoming connections to specific servers based on the target domain name. From Zorp version 7.0.12, the **HashResolver** policies also cache the domain names and the IP addresses found.

Example 6.16. Using HashResolver to direct traffic to specific servers
If a Zorp host is protecting a number of servers located in a DMZ, the connections can be easily directed to the proper server without a DNS query if the hostname – IP address pairs are stored in a HashResolver. If multiple IPs are associated with a hostname, simple fail-over functionality can be realized by using **FailOverChainer**.

The resolver policy below associates the IP addresses *192.168.1.12* and *192.168.1.13* with the *mail.example.com* domain name.


ZMC - Edit resolver rule

Edit resolver rule

Hostname: mail.example.com

Addresses
"192.158.55.55"
"192.168.55.56"

Description:

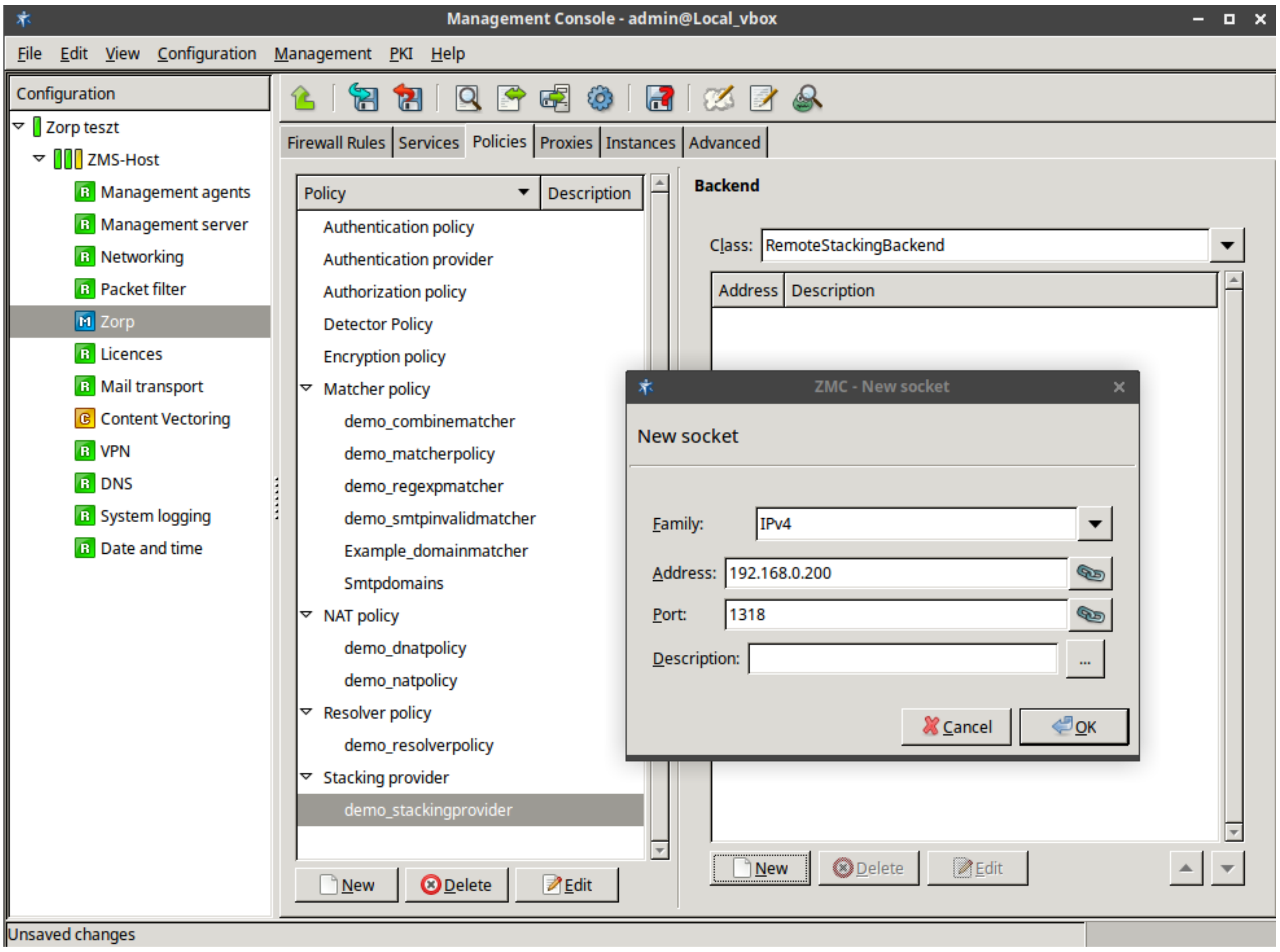
Figure 6.76. Defining a new HashResolver

```
Python:
ResolverPolicy(name="DMZ", resolver=HashResolver(mapping={"mail.example.com":\
("192.168.1.12", "192.168.1.13")}))
```

6.7.7. Stacking providers

Stacking providers are external hosts managed by the Zorp Content Vectoring System (ZCV) performing various traffic analysis and filtering (for example, virus and spam filtering). These hosts can be listed as Stacking providers, and easily referenced from multiple service definitions.

A Stacking provider includes the IPv4 or unix socket of the host performing the analysis of the traffic. If multiple hosts are set in a single policy, they will be used in a fail-over fashion, that is, if the first host is down, the traffic will be directed to the second one for analysis, and so on.



Class: RemoteStackingBackend

Family: IPv4
Address: 192.168.0.200
Port: 1318
Description:

Figure 6.77. Creating a new Stacking Provider through IPv4

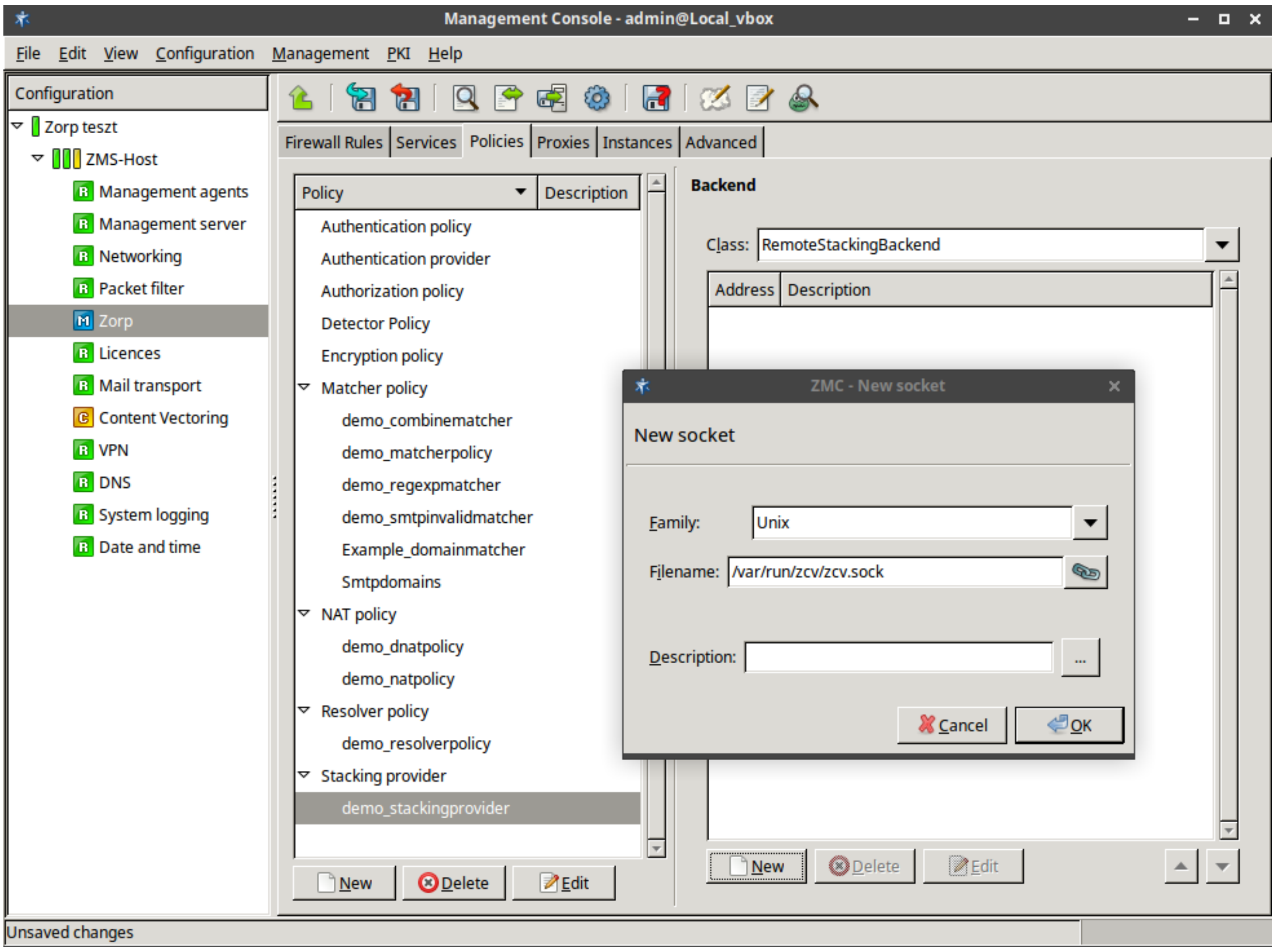


Figure 6.78. Creating a new Stacking Provider through domain socket

To specify an IPv4 socket, the IP address and the port of the host have to be specified. To specify a unix domain socket, the full path (including the actual filename) has to be provided.

For details about ZCV, its configuration and the use of stacking providers, see *Chapter 14, Virus and content filtering using ZCV (p. 351)*.

6.8. Monitoring active connections

ZMC provides a status window to monitor the active connections of an instance (for information on instances, see *Section 6.3, Zorp instances (p. 104)*). Navigate to the **Zorp** ZMC component, select an instance, then click **Active Connections**.

Tip
Multiple instances can also be selected.

The **Active connections** window displays the following parameters of the active services within the instance:

- **Name**: It provides the name of the service (for example, *`intra_HTTP_inter`*).
- **Proxy module**: It provides the name of the proxy (for example, *`MyHttpProxy`*).
- **Proxy class**: It identifies the proxy class from which the proxy module used in the service definition was derived (for example, `HttpProxy`).
- **Client address**: It is the IP address of the client.
- **Client port**: It is the port number of the client.
- **Client local**: It is the IP address targeted by the client.
- **Client local port**: It is the port targeted by the client.
- **Client zone**: It is the zone the client belongs to.
- **Server address**: It is the IP address of the server.
- **Server port**: It is the port number of the server.
- **Server local**: It is the IP address used by Zorp on the server-side (the server sees this address as client address).
- **Server local port**: It is the port used by Zorp on the server-side (the server receives the connection from this port).
- **Server zone**: It is the zone the server belongs to.

Note
The **Proxy class** and **Proxy module** parameters are empty for packet filter services.

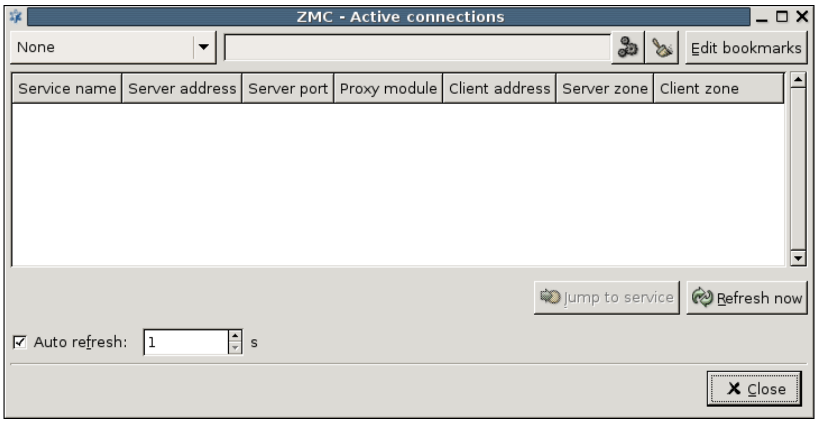


Figure 6.79. Active connections

The list of active connections is not updated real-time, it is only a snapshot. It can be updated by clicking **Refresh now**. The **Jump to service** displays the configuration of the selected service on the **Services** tab.

The **Active connections** window is a *Filter window*, thus various simple and advanced filtering expressions can be used to display only the required information. For details on the use and capabilities of *Filter windows*, see *Section 3.3.10, Filtering list entries (p. 48)*.

6.9. Traffic reports

Zorp can automatically create daily, weekly, and monthly statistics about the transmitted traffic, and send them to an administrator or auditor through e-mail. The reports are in Adobe Portable Document (PDF) format. Note that these reports do not provide detailed statistics about every host on your network, rather they can be used to identify the most active hosts ("top-talkers") and to examine trends and sudden changes in the statistics (outliers).

In general, every section of the report consists of a table that details the ten most active clients (for example, the ten clients who transferred the most data in a zone) and a pie chart that displays every client. Note that on the pie chart, only the clients responsible for at least ten percent of the total value are labeled, all other clients are aggregated under the `Others` label.

Note
Every Zorp host, and every node of a Zorp cluster creates and sends a separate report. Reporting options must be configured on every Zorp host separately.

The reports include the following information:

- **Network Traffic**: It provides traffic statistics for the entire network.
- **Zone Traffic**: It provides traffic statistics for every zone defined in Zorp. Note that this report can be long if there are many zones defined.
- **Mail Delivery Traffic**: It provides statistics for the total transferred SMTP traffic, as well as for the most active accounts. Top senders and recipients are listed separately.
- **Spam and Virus Reports**: It provides statistics about spam and infected e-mails.
- **Access Control Reports**: It provides statistics about connection-attempts that were blocked by Zorp.
- **URL Reports**: It provides a list of websites generating more than a set amount of traffic, and a list of their top visitors. By default, ULR Reports are not included in the regular reports, see *Procedure 6.9.1, Configuring Zorp reporting (p. 188)* for details on configuring them.

6.9.1. Procedure – Configuring Zorp reporting

Step 1. Login to the Zorp host locally, or through SSH, and edit the `/etc/zorp/reports/options.conf` file. Alternatively, you can add this file to the Text Editor plugin of ZMC (see *Procedure 8.1.1, Configure services with the Text editor plugin (p. 213)* for details).

Step 2. Enter the e-mail address to the *`ADMINEMAIL`* field, where the reports should be sent to.

Step 3. Enter a name for the daily, weekly, and monthly report files into the *`TITLE_DAILY, TITLE_WEEKLY, TITLE_MONTHLY`* fields.

Step 4. By default, the reports do not include statistics about visited websites. To include statistics about visited websites and the clients who visited them, enter a positive number into the *`URLS`* field. URLs that generate traffic higher than this value in megabytes will be included in the reports. For example, *`URLS=1024`* will include every website that generates at least 1 GB traffic. The direction of the traffic does not matter, uploads and downloads are considered together.

Step 5. Save the file.

Warning
The reports are only sent to the e-mail address set in `/etc/zorp/reports/options.conf`, they are not stored locally on the Zorp host.

Tip
To generate reports manually for arbitrary time periods, use the `report.py` command-line tool. Run `report.py` without parameters to display the available options and parameters of the application.

Chapter 7. Logging with syslog-ng

Firewall design and implementation generally require the presence of a subsystem that is responsible for accountability. This subsystem stores information on user and administrator activities performed on or through the firewall and must be configurable to provide exactly the amount and type of information needed, which, on the other hand, is usually defined by the corporate IT Security policy.

This requirement in computer systems is typically fulfilled by a logging subsystem. Logging is the act of collecting information about events in the system. The result of logging can be the following.

- one or more log files (binary or text-based files)
- the console
- rows in a database table if logging is set up to use a database as the output store

These log files are archived for event-tracking purposes. Apart from a simple, automated archiving procedure, however, these log files must be continuously analyzed. Logging records, (quasi) real-time user and system activities, that is, a continuous analysis, is capable of detecting malicious user activities or system malfunctions as they take place, thereby allow for an effective and quick response policy.

In security-sensible systems, such as the Zorp firewall, logging must be a system–wide action, that is, all the relevant components of the system provide logging information, or log entries. There must be therefore a central component, a logging subsystem, that collects log entries from the various system components and organizes them into a log file of some format (the term file is used literally here and does not necessarily mean a text or binary file: it is simply the output of logging that can be a database, the console or some input queue on another machine). It would not be practical nor economical for all the components to manage their own log files. It would waste system resources (more open file handles, database connections or network sessions, depending on the output destination used) and would make analysis cumbersome (hunting for information in several separate log files that would probably be syntactically incompatible).

In Zorp, centralization is elevated to a higher level: the logs of the central logging subsystem on all machines are unified on a network-wide dedicated central logging machine. This way, log analysis, reporting and archiving is simpler even for larger networks with many entities providing logging capabilities.

These requirements have, in part, long been targeted by the Unix syslog subsystem. For decades, syslog was the ultimate central logging subsystem for Unix and Linux machines. It is an old and rather unreliable technology and lacks some flexibility and security features that are required in today's demanding network and security environments. Therefore, a replacement service called syslog-ng, where "ng" stands for "next generation", has been created. The syslog-ng application inherited all the concepts, features and most of the naming conventions from syslog and enhanced it with several new features targeting flexibility and security requirements. Zorp uses syslog-ng as its logging subsystem.

The following sections include a short introduction to the concepts of syslog-ng, and describe how ZMS works with syslog-ng and how this subsystem can be managed with ZMC.

7.1. Introduction to syslog-ng

The syslog-ng application runs as a daemon process and collects information from various log sources. Depending on the options and filters configured, syslog-ng saves the received log entries to the specified destinations. The configuration of syslog-ng mainly consists of configuring its components correctly.

The components of syslog-ng are the following:

- Sources
- Global options
- Filters
- Destinations

The syslog-ng configuration is stored in a text-based configuration file that is typically the `/etc/syslog-ng/syslog-ng.conf` file. ZMC hides the exact structure of this configuration file and takes care of the correct syntax, allowing the administrator to concentrate on the actual configuration tasks. However, as syslog-ng is present in more and more Linux/Unix distributions, it is beneficial to know the syntax and the content of this configuration file too. In addition, syslog-ng allows for centralized logging from machines not necessarily under the control of ZMS. In this case configuring syslog-ng means manually editing the corresponding configuration file.

The `syslog-ng.conf` file has a C-like syntax with curly braces (`{}`) separating integral code parts and with semicolons (`;`) for closing expressions. Comments begin with hashmark (#).

7.1.1. Global options

It is possible to provide syslog-ng with global options that affect all the logging commands, although they can be overridden, if required. For further details, see *Section 7.2.2.3, Configuring destinations (p. 202)*.

There are several different options available, some of them are especially useful when dealing with log entries coming from other machines on the network. By default, these entries are recorded using the sender host's IP address, but by using options *`use-fqdn()`*, *`use-dns()`*, *`chain-hostnames()`* and *`keep-hostname()`* it is also possible to look up the hostnames for servers generating log entries.

For a full list of the available options and their descriptions, see *Appendix C, Further readings (p. 485)*.

7.1.2. Sources

There are several system components that do not output log entries in a unified format or method. Some of them output to files, while others use a pipe, or use a unix-stream. Some can even be configured to use a certain output method. The syslog-ng application can accept log entries from these output methods too.

The syslog-ng application supports the following source types:

- *`internal()`*
  The log messages of syslog-ng itself.

- *`file()`*
  This source is for log entries from a special file, like `/proc/kmsg`.

Note
A file source cannot be an ordinary text file, for example, one generated by `httpd`. However, it is possible to feed syslog-ng with messages from such a file indirectly. For this, a custom script is required, for example, a script that uses `tail -f` to transfer messages from the desired logfile to the logger utility.

- *`pipe()`*
  This source is for messages from a pipe.

- *`unix_stream()`*
  This source is for log entries from a connection–oriented socket.

- *`unix_dgram()`*
  This source is for log entries from connectionless sockets.

- *`tcp()`*
  Log entries from remote machines that use TCP for log entry submission.

  Note
  One of the advantages of syslog-ng over traditional syslog is that it can handle TCP connections.

  By default, syslog-ng uses TCP port `514`.

- *`udp()`*
  Log entries for remote machines that use UDP for log entry submission.

  By default, syslog-ng uses UDP port `514`.

- *`systemd-journal()`*
  This source is for collecting messages from the systemd-journal system log storage.

The most important sources when dealing with local component's log entries are probably *`unix_stream()`* and *`unix_dgram()`*, because the main system components, like the kernel and many of the daemon processes as well use one of them for recording log events.

7.1.3. Destinations

The syslog-ng application can send log messages to the following types of destinations.

- *`file()`*
  This destination is an ordinary text file. It is possible to use macros in filenames, thus you can create dynamic file names.

- *`pipe()`*
  This destination is a named pipe.

- *`program()`*
  This destination means the standard input of a given program.

- *`syslog()`*
  Send messages to a remote syslog server specified by its IP address or FQDN using the *RFC 5424 (IETF syslog)* protocol over TLS, TCP, or UDP. The default destination port is `514`.

- *`tcp()`*
  Send messages to a remote syslog server specified by its IP address or FQDN using the *RFC 3146 (BSD syslog or legacy-syslog)* protocol over TCP or TLS. The default destination port is `601`.

- *`udp()`*
  Send messages to a remote syslog server specified by its IP address or FQDN using the *RFC 3146 (BSD syslog or legacy-syslog)* protocol over UDP. The default destination port is `514`.

- *`unix_dgram()`*
  This destination is a connectionless Unix socket destination.

- *`unix_stream()`*
  This destination specifies a connection–oriented Unix socket as a destination for log entries, for example, `/dev/log`.

- *`usertty()`*
  This destination sends log messages to the terminal of a given user. Username is given as a parameter of usertty.

7.1.4. Filters

To fine-tune what log entries are needed for or how they are forwarded to different destinations, it is possible to use filters in syslog-ng configurations. Although their usage is optional, they are highly recommended because they represent the real flexibility of syslog-ng.

Filtering can be defined to use seven different criteria that are summarized in the following list.

facility()	It filters the type of messages referring to the nature of the log entry. For example, `auth`, `cron`, `daemon`, `kern`, `mail`.
priority()	It filters the assigned priority level of the log message. The possible priority levels are the following in the order of severity: `none`, `debug`, `info`, `notice`, `warning`, `err`, `crit`, `alert`, `emerg`.
level()	It is the same as priority.
program()	It is the name of the software component that generated the log entry.
host()	It is the machine that the log message arrived from.
match()	It is a regular expression that is compared to the contents of the log message.
filter()	It is an additional filter.

By combining these elements you can manually configure a fairly complex logging environment in a couple of lines of “code”, with basic knowledge on the syntax of syslog-ng rules. If you use ZMC, ZMC takes care of the correct syntax and allows you to focus on the actual rule creation process.

For more detailed information on syslog-ng, see *Appendix C, Further readings (p. 485)*.

7.2. Configuring syslog-ng with ZMC

7.2.1. Procedure – Configure syslog-ng

To configure system logging in Zorp Professional complete the following steps.

Step 1. Select the host where you want to configure system logging, then click **New**.

Step 2. Choose a template for the **System logging** component.

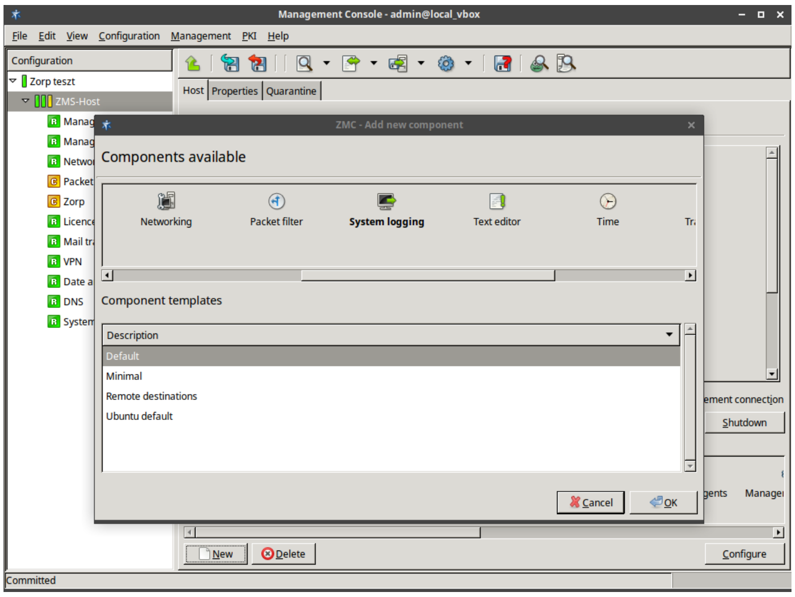
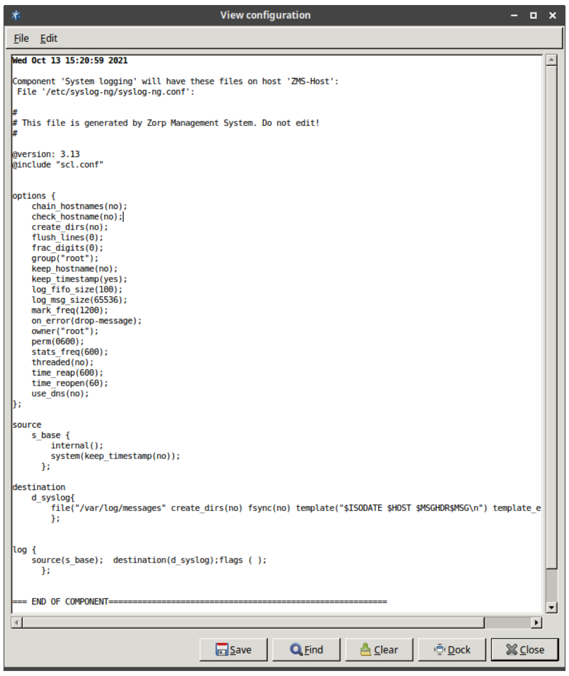


Figure 7.1. Selecting a syslog-ng template

The following templates are available for the component:

- **Default**: It collects logs from `/dev/log` and `/proc/kmsg`, and stores them in the `/var/log/messages` file.
- **Remote destinations**: Type the IP address of your logserver into the **Logserver IP** field. It collects logs from `/dev/log` and `/proc/kmsg`, and stores them in the `/var/log/messages` file. The log messages are also sent to the logserver over TCP using the legacy BSD-syslog protocol (RFC3164).
- **Debian default**: It collects logs from `/dev/log` and `/proc/kmsg`, and stores them in several different files, like the default syslog configuration on Debian systems.
- **Minimal**: It defines an empty configuration.

Step 3. View this initial configuration file by selecting the system logging component of a given host and click the **View Current Configuration** button.

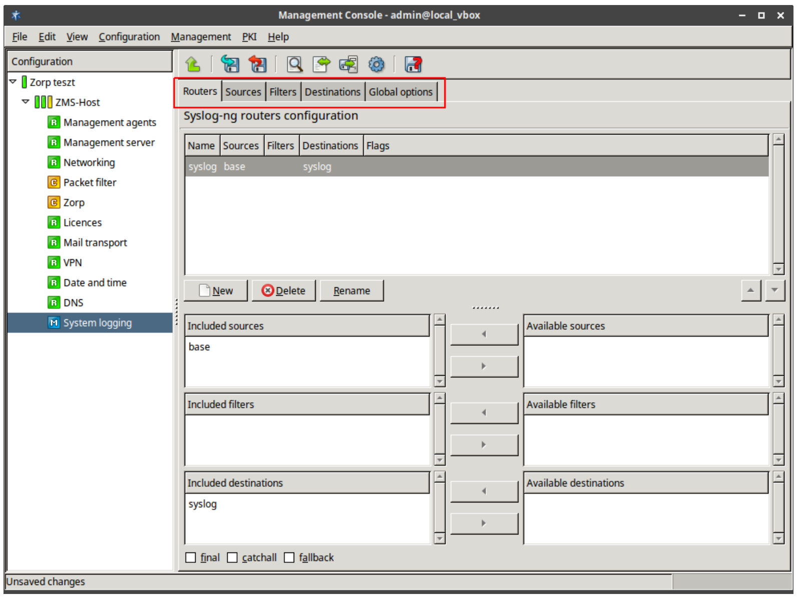


```
Wed Oct 13 15:20:59 2021

Component 'System logging' will have these files on host 'ZMS-Host':
 File '/etc/syslog-ng/syslog-ng.conf':

#
# This file is generated by Zorp Management System. Do not edit!
#

@version: 3.13
@include "scl.conf"


options {
    chain_hostnames(no);
    check_hostname(no);
    create_dirs(no);
    flush_lines(0);
    frac_digits(0);
    group("root");
    keep_hostname(no);
    keep_timestamp(yes);
    log_fifo_size(100);
    log_msg_size(65536);
    mark_freq(1200);
    on_error(drop-message);
    owner("root");
    perm(0600);
    stats_freq(600);
    threaded(no);
    time_reap(600);
    time_reopen(60);
    use_dns(no);
};

source
    s_base {
        internal();
        system(keep_timestamp(no));
      };

destination
    d_syslog{
        file("/var/log/messages" create_dirs(no) fsync(no) template("$ISODATE $HOST $MSGHDR$MSG\n") template_e
        };


log {
    source(s_base);  destination(d_syslog);flags ( );
      };


=== END OF COMPONENT==========================================================
```

Figure 7.2. Basic `syslog-ng.conf` file created from the system logging chroot template

7.2.2. Configuring syslog-ng components through ZMC

The main configuration window of the system logging component consists of five tabs, corresponding to the five main components of a syslog-ng configuration.



Syslog-ng routers configuration

Name	Sources	Filters	Destinations	Flags
syslog	base		syslog	

Included sources	Available sources
base	

Included filters	Available filters

Included destinations	Available destinations
syslog	

Figure 7.3. Configuration tabs for the system logging component

The only tab that needs some explanation is the first one, **Routers**. This tab is used to assemble the actual log commands from the other components, so a router actually represents a log command entry in the `syslog-ng.conf` file. Therefore, during a configuration cycle, it is recommended to visit and configure the **Routers** tab last. First, configure the **Global options** tab to set up system-wide defaults, then you can continue with the **Sources** and **Destinations** tab.

7.2.2.1. Configuring global options

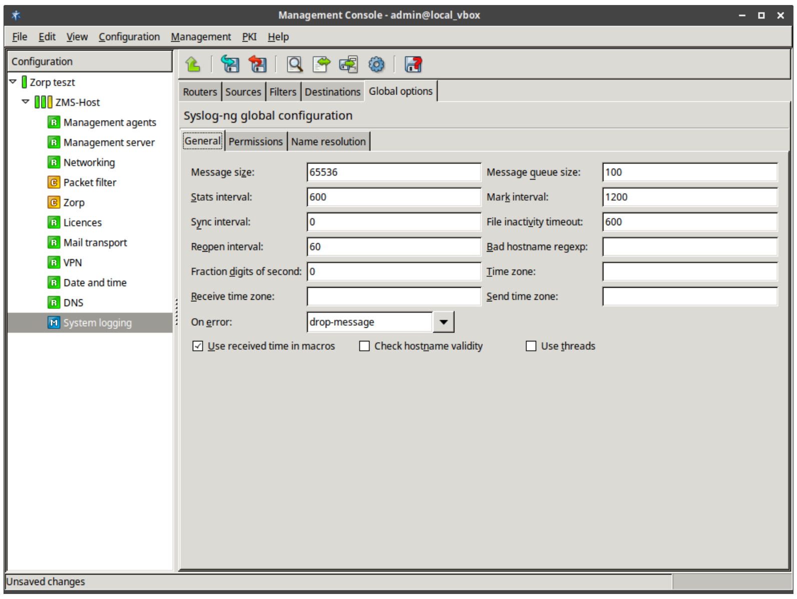
7.2.2.1.1. Procedure – Set global options

The *Global options* main tab contains three further sub-tabs for configuring the necessary parameters:

- General
- Permissions
- Name resolutions

Step 1. Configure the parameters for I/O operation optimization.
File I/O is always expensive in terms of system time needed, so theoretically the number of (log) write operations should be minimized, keeping a number of incoming log entries in a memory buffer and batch-write them out to disk.

> **Note**
> This buffer and thus the time between successive log write-outs shall not take too long because in case a hardware malfunction occurs and the machine has to be rebooted, the log messages that have not been written out yet are lost.



Message size:	65536	Message queue size:	100
Stats interval:	600	Mark interval:	1200
Sync interval:	0	File inactivity timeout:	600
Reopen interval:	60	Bad hostname regexp:	
Fraction digits of second:	0	Time zone:	
Receive time zone:		Send time zone:	
On error:	drop-message		

[✓] Use received time in macros ☐ Check hostname validity ☐ Use threads

Figure 7.4. Global syslog-ng options for file handling

Time-related parameters are given in seconds, message size is in bytes, while message queue size is an item number.

Step 2. Set system time usage.

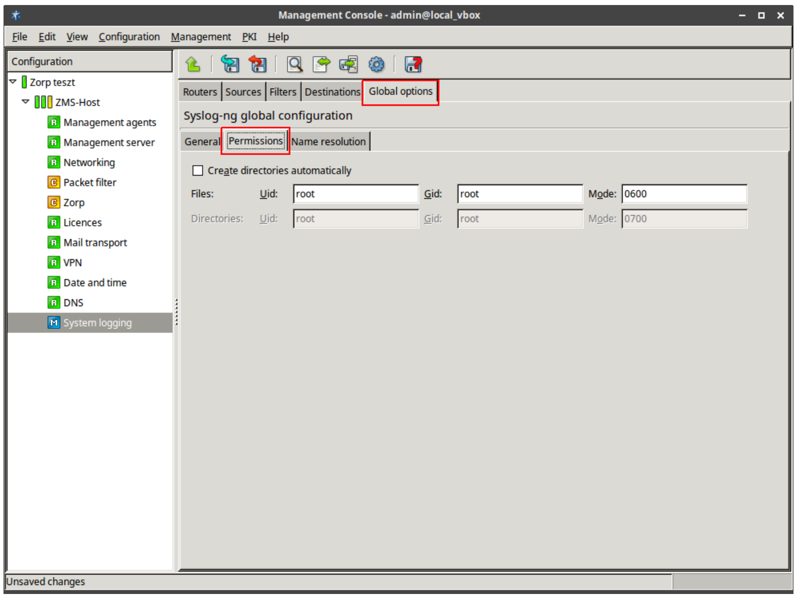
Macro substitution is possible in syslog-ng, for example when creating filenames. If you use system time as a macro variable, the default is to use local system time on the syslog-ng server that processes the log entries. If, instead, you want to use time values received in the log messages themselves, check the **Use received time in macros** checkbox.

Step 3. Configure the required parameters under *General* tab.
The list of other configurable parameters in this tab includes the following.

Message size	It defines the allowed maximum size for log messages.
Message queue size	It defines the allowed number of messages waiting to be processed.
Stats interval	It sets the syslog-ng's internal reporting interval. The syslog-ng application reports a number of parameters on its own operations and statistics.
Mark interval	It sets the regularity of marking timestamps by the syslog daemon.
Sync interval	It defines how often log messages are written out from memory. The default '0' means there is no time delay, messages are written out continuously.
File inactivity timeout	It defines how long after the non-usage time the log files are closed.
Reopen interval	It sets how often a log file can be opened again.
Bad hostname regexp	This is a regexp which contains hostnames that should not be handled.
Fraction digits of second	The syslog-ng application can store fractions of a seconf in the timestamps according to the ISO08601 format. This parameter specifies the number of digits stored.
Time zone	By setting this parameter timestamps will be converted to the timezone specified here. This timezone will be associated with the messages only if no timezone is specified within the message itself.
Receive time zone	It specifies the time zone associated with the incoming messages, if it is not specified otherwise in the message or in the source driver.
Send time zone	It specifies the time zone associated with the messages sent by syslog-ng, if it is not specified otherwise in the message or in the destination driver.
On error	It controls what happens when type-casting fails and syslog-ng cannot convert some data to the specified type.
Use received time in macros	It specifies whether syslog-ng shall accept the timestamp received from the application or client

	sending it. If it is disabled, the time of reception will be used instead.
Check hostname validity	A check whether the hostname contains valid characters or not can be enabled or disabled.
Use threads	This parameter enables multithreading in syslog-ng.

Step 4. Assign owner and permission parameters on the *Permissions* tab to log files and directories created by syslog-ng.



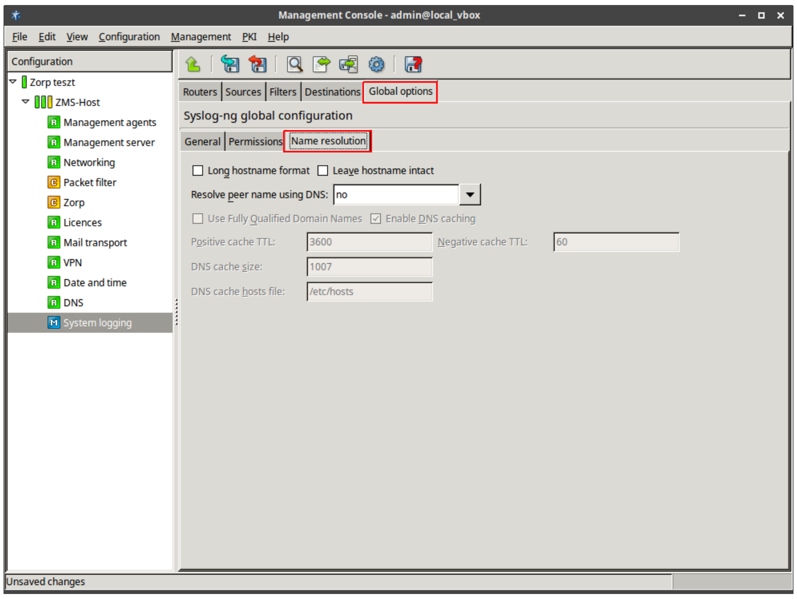
☐ Create directories automatically

Files:	Uid:	root	Gid:	root	Mode:	0600
Directories:	Uid:	root	Gid:	root	Mode:	0700

Figure 7.5. Permission settings for logfile creation

By default, syslog-ng runs as root, but can be configured to run as a limited user as well. In this case you have to set the appropriate permissions, or use the default values.

Step 5. Set name resolution for syslog-ng under the *Name resolutions* tab.



Syslog-ng global configuration

☐ Long hostname format ☐ Leave hostname intact

Resolve peer name using DNS: no

☐ Use Fully Qualified Domain Names ☑ Enable DNS caching

Positive cache TTL: 3600 Negative cache TTL: 60

DNS cache size: 1007

DNS cache hosts file: /etc/hosts

Figure 7.6. Name resolution settings for syslog-ng

Machine identification in log entries is accomplished by using IP addresses. If you want to use hostnames that are easier to remember and recognize, you can instruct syslog-ng to perform name resolution. This name resolution only works for resolving the IP addresses of hosts sending log entries.

If there are IP addresses within the log messages themselves, they are not resolved this way. To perform name resolution for those addresses, a log analyzer utility is needed. Name resolution is a time-consuming process and to achieve the best results, use a DNS server that is “close” to the syslog-ng server in terms of response time.

On the other hand, log entries are typically coming from a limited number of machines (servers) and their IP addresses tend not to change. Therefore, it is reasonable for the syslog-ng server to cache their resolved names locally, thus easing the heavy reliance on a DNS server.

You can configure DNS caching as a global option, under the name resolution tab. The time values are in seconds, cache size is in bytes. File options can be changed in individual file destination configurations, but name resolution options cannot, they are always global.

7.2.2.2. Configuring sources

Sources are collections of communication channels where log entries can arrive. A source in syslog-ng can consist of one or more drivers. Driver is the actual communication channel that must be monitored for log messages.

By default, using one of the default templates, there is an internal driver for syslog-ng's own log messages, and there are three unix-stream drivers for `/dev/log`, `bind`, and `ntp`, respectively.

The default source is called base. To configure new drivers, either define them under this default source or create new sources. A source must always contain at least one driver.

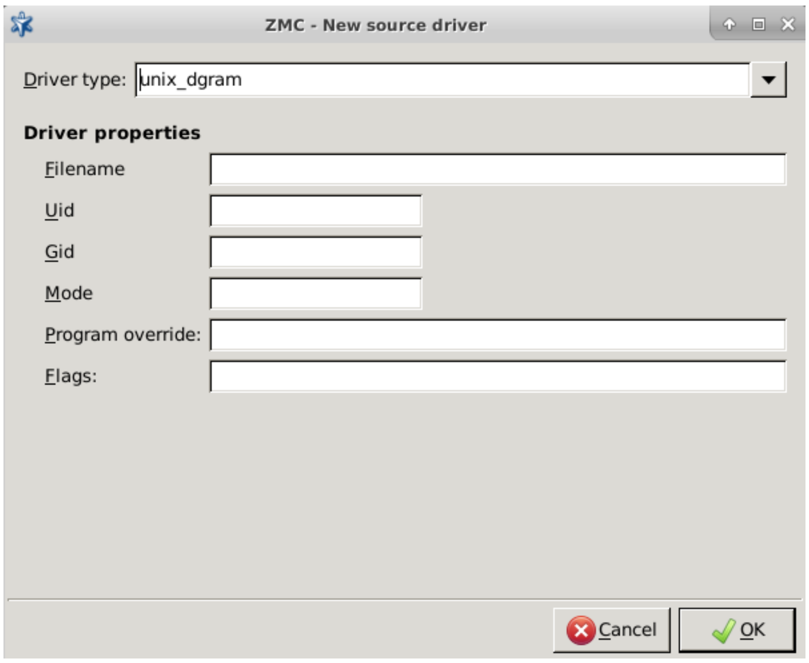
7.2.2.2.1. Procedure – Create sources

Step 1. Create a source click on **New**.

Step 2. Provide a name for the source.

7.2.2.2.2. Procedure – Create drivers

Step 1. Click **New** on the **Drivers** subwindow on the **Sources** tab in `System logging` component.
The following window appears.



Driver type: unix_dgram

Driver properties

Filename

Uid

Gid

Mode

Program override:

Flags:

Figure 7.7. Adding a new source driver for syslog-ng

Step 2. Select a driver type.
The rest of the options are based on this selection.

Step a. For unix_dgram, unix_stream, sun_stream and file driver types, set the filename.

Note
None of these driver types are ordinary text files. This file is a binary file while the others are socket endpoints. Nevertheless, they are identified by filenames.

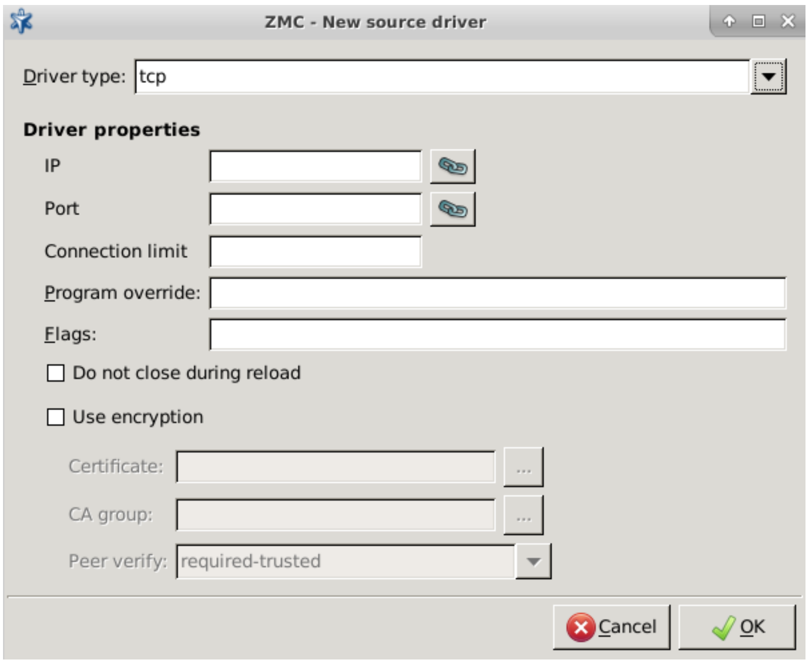
Step b. If you have a custom system component, for example, a daemon, that sends its log messages to a special socket and you want syslog-ng to collect this component's log messages, set up a driver for it. Many of the Linux daemons and other software components prefer `/dev/log` but it is not a central requirement. Some software applications can even be instructed with the help of the configuration file where to log.

Step c. For TCP and UDP source drivers, specify an IP address and a port number.

The machine running syslog-ng waits for log messages from other servers on this IP address/port pair. In other words, here you do not specify from where, that is, what machines the log entries arrive from, but rather on what IP address/port pair syslog-ng collects these log entries.

The default port for both TCP and UDP is 514.

For TCP drivers some additional parameters can be supplied.


ZMC - New source driver

Driver type: tcp

Driver properties

IP

Port

Connection limit

Program override:

Flags:

☐ Do not close during reload

☐ Use encryption

Certificate:

CA group:

Peer verify: required-trusted

Figure 7.8. Configuring TCP source drivers

Since TCP is a connection-oriented protocol, a virtual session is always established between the communicating parties. This session buildup takes time and bandwidth (three-way handshake), therefore to save some of these resources, if a session is built between syslog-ng and the host sending log entries, it is kept alive with the help of **keep alive messages**. However, if the number of active TCP sessions is high, it can have negative effect on the performance of the host running syslog-ng. On the other hand, if the number of sessions is kept low, using the **Connection limit** setting, some log messages may be lost if the connection limit has already been reached.

The **Program override** parameter enables replacing the `${PROGRAM}` part of the message with the provided parameter string. The **Flags** parameter specifies the log parsing options of the source.

Another small optimization setting is the **Do not close during reload** checkbox: it instructs the system not to close open TCP sessions while syslog-ng configuration is reloaded.

These two settings are available for the unix_stream driver type as well.

Additional parameter configuration options are as follows:

- Use ancryption: If this option is enabled, a TLS-encrypted channel is used.
- Certificate: It specifies the certificate used to authenticate the syslog-ng client on the destination server.
- CA group: It specifies the CA group to verify peer certificates.
- Peer verify: This option defines the verification method of the peer.

7.2.2.3. Configuring destinations

The logic behind destination configuration is the same as with sources' configuration. You can create one or more destination directives and fill them with drivers specifying the actual channels on which log messages are recorded.

The available destinations are the following:

- file
- syslog: TCP, UDP, or TLS-encrypted TCP through the RFC5424 (IETF-syslog) protocol
- TCP through the RFC3164 (BSD-syslog or legacy-syslog) protocol, including TLS-encrypted TCP
- UDP through the RFC3164 (BSD-syslog or legacy-syslog) protocol
- pipe
- program
  Program specifies the input of a software, typically a script, that can perform an action based on the input log message it receives.

  Tip
  This may be a good solution to set up an alerting mechanism.

- Unix_dgram
- Unix_stream
- Usertty
  Usertty is a terminal to which log messages can be displayed; it is meaningful as a destination if there is someone actually monitoring that terminal, or the named terminal is routed to some other, special destination.

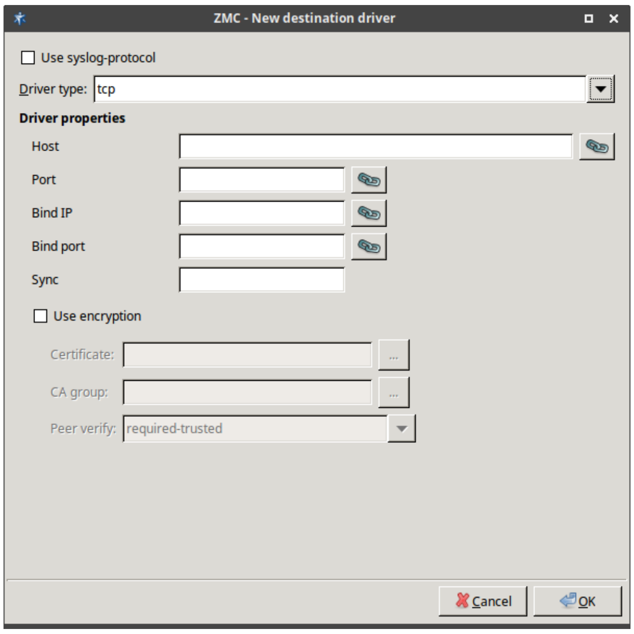


Figure 7.9. Configuring TCP and UDP destinations

Host and **Port** define the destination of log messages on the network, while **Bind IP** and **Bind port** specify where (in which direction) the log messages are sent out from the host. This is especially important for firewalls which almost always have two or more interfaces and a number of IP addresses.

Additional parameter configuration options are as follows:

- Use ancryption: If this option is enabled, a TLS-encrypted channel is used.
- Certificate: It specifies the certificate used to authenticate the syslog-ng client on the destination server.
- CA group: It specifies the CA group to verify peer certificates.
- Peer verify: This option defines the verification method of the peer.

File destination also has some important properties that need some explanation:

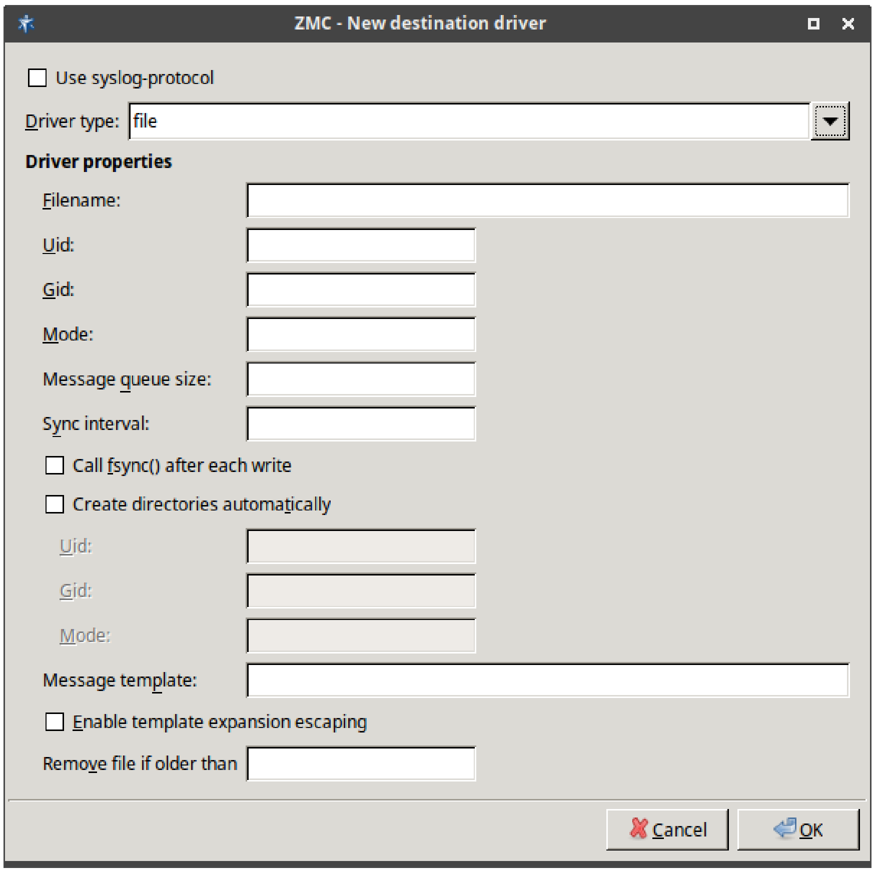

ZMC - New destination driver
Use syslog-protocol
Driver type: file
Driver properties
Filename:
Uid:
Gid:
Mode:
Message queue size:
Sync interval:
Call fsync() after each write
Create directories automatically
Uid:
Gid:
Mode:
Message template:
Enable template expansion escaping
Remove file if older than
Cancel
OK

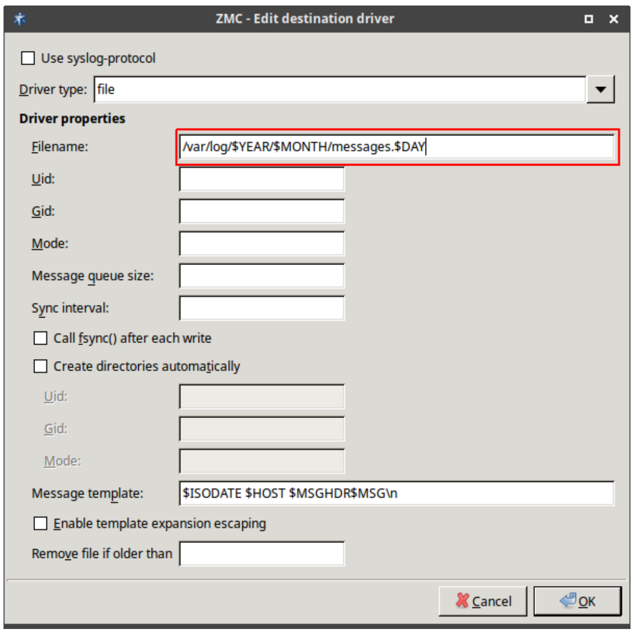
Figure 7.10. Configuring a file destination driver

Most properties are present also on the **Global options** tab: many of the global options can be overridden in file destination setup. For the descriptions of the properties, see section *Section 7.1.1, Global options (p. 191)*.

A special property is the **Message template**. This property can be used to apply some basic formatting on log messages: according to the example given in *Figure 7.11, Macro substitution in file naming (p. 205)*, all log messages that end up in `/var/log/syslog` have a timestamp (`$ISODATE`) and a hostname (`$HOST`) inserted before the actual log message (`$MSG`). This is the default behavior in Zorp.

Note
The **Message template** property is not to be confused with macro substitution in filename creation. Message templates can format actual log messages, while macro substitution can format log file names.

For example, if you want to create a new logfile every day, modify the filename property in *Figure 7.11, Macro substitution in file naming (p. 205)* the following way.



☐ Use syslog-protocol

Driver type: file

Driver properties

Filename: /var/log/$YEAR/$MONTH/messages.$DAY

Uid:

Gid:

Mode:

Message queue size:

Sync interval:

☐ Call fsync() after each write

☐ Create directories automatically

Uid:

Gid:

Mode:

Message template: $ISODATE $HOST $MSGHDR$MSG\n

☐ Enable template expansion escaping

Remove file if older than

Figure 7.11. Macro substitution in file naming

7.2.2.4. Configuring filters

An optional component of syslog-ng configuration is filter creation. Filters can be used to pick log entries from defined sources with the possible intent of sending selected log entries to different destinations.

Example 7.1. Selecting log messages from Postfix using filter
The following is a trivial filter to select log messages coming from Postfix:

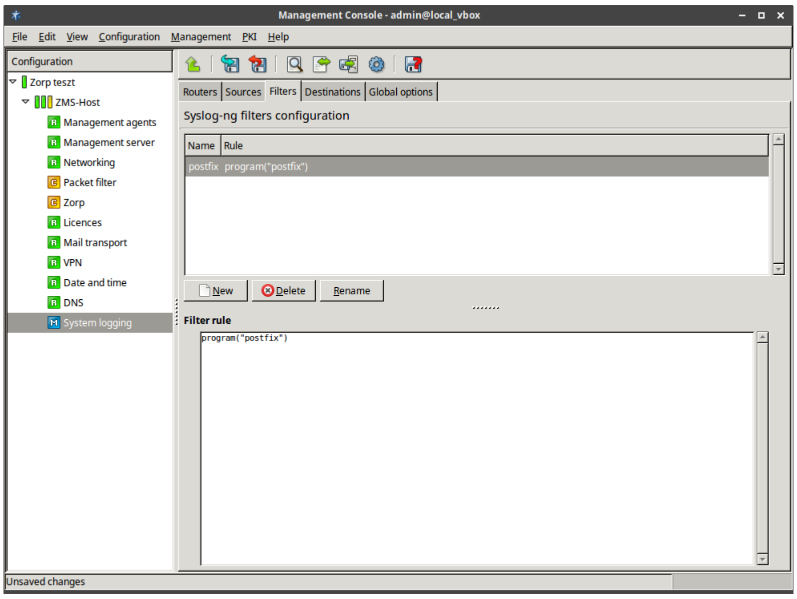
```
filter f_postfix{program(“postfix”);};
```

Filters can use regular expressions in a match criteria and a number of other criteria as well. For a complete list of criteria, see *Section 7.1.4, Filters (p. 193)*. Due to the flexible nature of filters, it is almost impossible to create a usable GUI to interface them. Therefore, the **Filter** tab of the `System logging` component is quite simple.

7.2.2.4.1. Procedure – Set filters

Step 1. Create one or more filters.
See *Section 7.2.2.2, Configuring sources (p. 199)* and *Section 7.2.2.3, Configuring destinations (p. 202)*.

Step 2. Set up a rule for each filter in the **Filter rule** textbox.



Name	Rule
postfix	program("postfix")

```
program("postfix")
```

Figure 7.12. The Filter rule textbox

ZMC aids in filter creation by taking care of the necessary curly braces ({}) and semicolons (;).

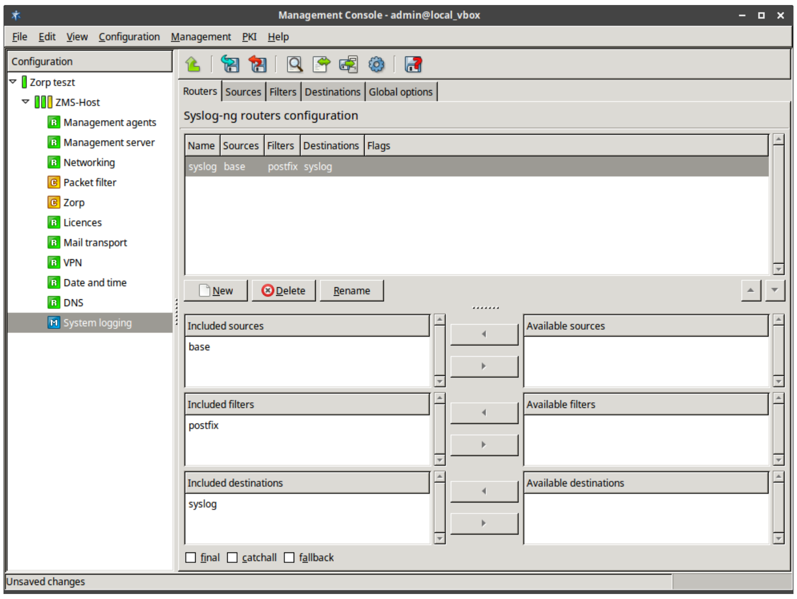
To create a syntactically correct postfix filter, enter the following details to the filter rule textbox:

`program(“postfix”)`.

For further information on possible filters, see *Appendix C, Further readings (p. 485)*.

7.2.2.5. Configuring routers

Logging rules are called Routers in syslog-ng terminology. Rules consist of a source, optionally a filter and a destination. The **Routers** tab of the `System logging` component represents this philosophy well.



Syslog-ng routers configuration

Name	Sources	Filters	Destinations	Flags
syslog	base	postfix	syslog	

Included sources: base

Included filters: postfix

Included destinations: syslog

Figure 7.13. Configured routers

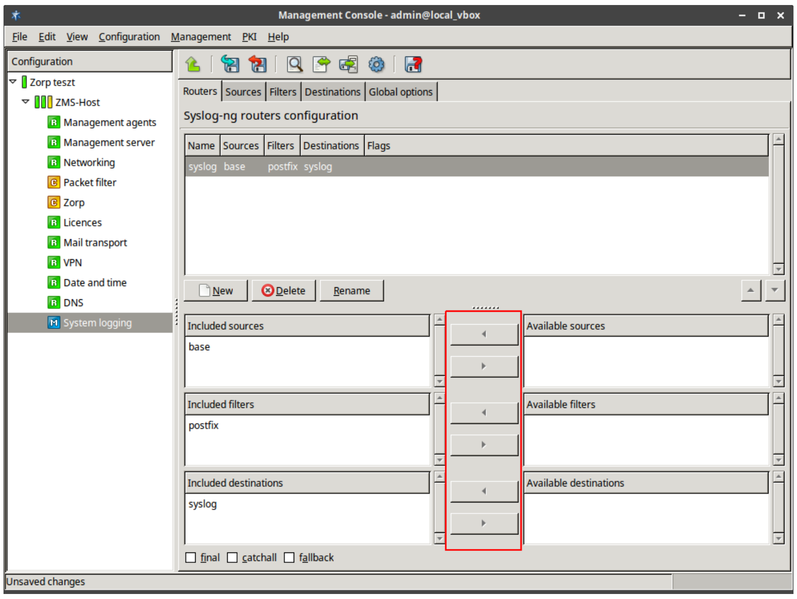
Just like sources, destinations and filters, more than one router can be present in the system. If you use several routers, it is recommended to apply a good naming strategy to easily identify the relevant log rules.

7.2.2.5.1. Procedure – Configure routers

Step 1. To create a router click on **New**.

Step 2. Provide a name for the new router.

Step 3. Select the components from the list of available sources, filters and destinations.



Name	Sources	Filters	Destinations	Flags
syslog	base	postfix	syslog	

Included sources: base

Included filters: postfix

Included destinations: syslog

Figure 7.14. Selecting router components

Example 7.2. Setting up a router
If you select `base` as the source, `postfix` as the filter (optional, use the small arrow between the text boxes) and `syslog` as the destination, the resulting router, that is, the log rule looks like this:

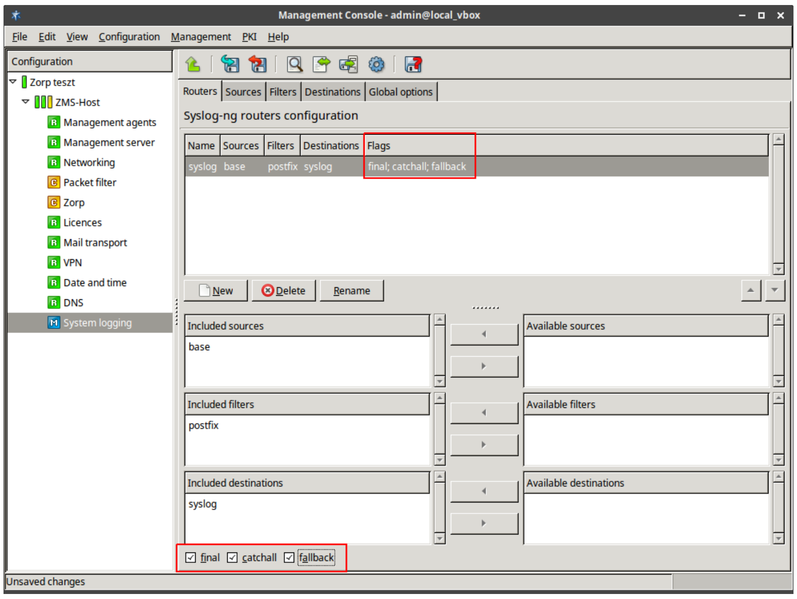
```
log { source(s_base); filter(f_postfix); destination (d_syslog);flags ( ); };
```

Step 4. Specify flags for the router.
The flags component is empty. Use the checkboxes at the bottom to select the flag. Three possible flags are available.

- Final flag means that the processing of log statements ends at this point.

  Note
  Ending the processing of log statements does not necessarily mean that matching messages are stored once, as there can be matching log statements processed prior to the current one.

- Fallback flag marks a log statement for 'fallback'. Defining a fallback statement means that only those messages are sent which do not match any 'non-fallback' log statements.
- Catchall flag means that the source of the message is ignored, only the filters are taken into account when matching messages.



Syslog-ng routers configuration

Name	Sources	Filters	Destinations	Flags
syslog	base	postfix	syslog	final; catchall; fallback

Included sources: base

Included filters: postfix

Included destinations: syslog

☑ final ☑ catchall ☑ fallback

Unsaved changes

Figure 7.15. Configuring flags for routers

There are virtually endless possibilities for configuring a complex system logging architecture with syslog-ng. This chapter focused only on the basic concept and provided an architecture view including not only Zorp and the ZMS host nodes, but presenting as well that the syslog-ng architecture can also include practically Unix/Linux machines.

For further information and details, see *The syslog-ng Administrator Guide*.

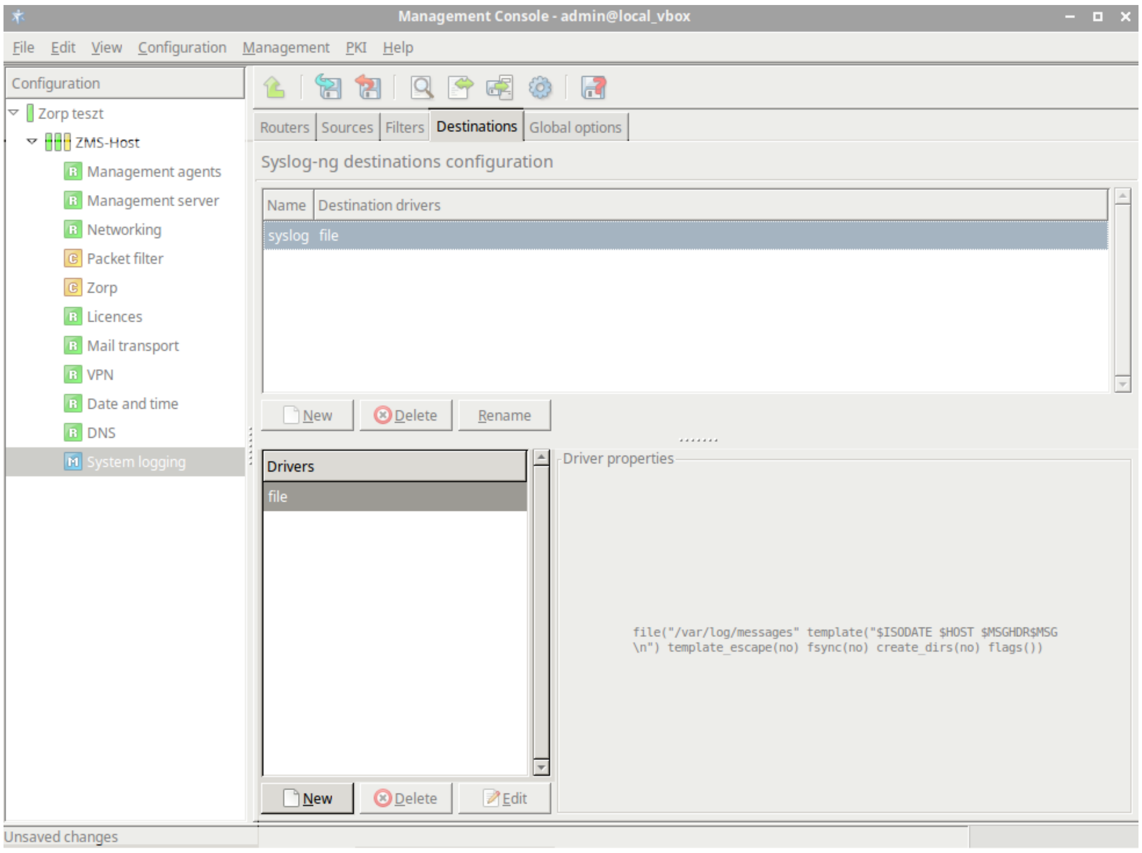
7.2.3. Procedure – Configuring TLS-encrypted logging

Purpose:

To encrypt the communication between the Zorp host and your central syslog server, complete the following steps.

Steps:

Step 1. Navigate to **System logging** > **Destinations** > **New**, and enter a name for the new destination (for example, `tls-logserver`).



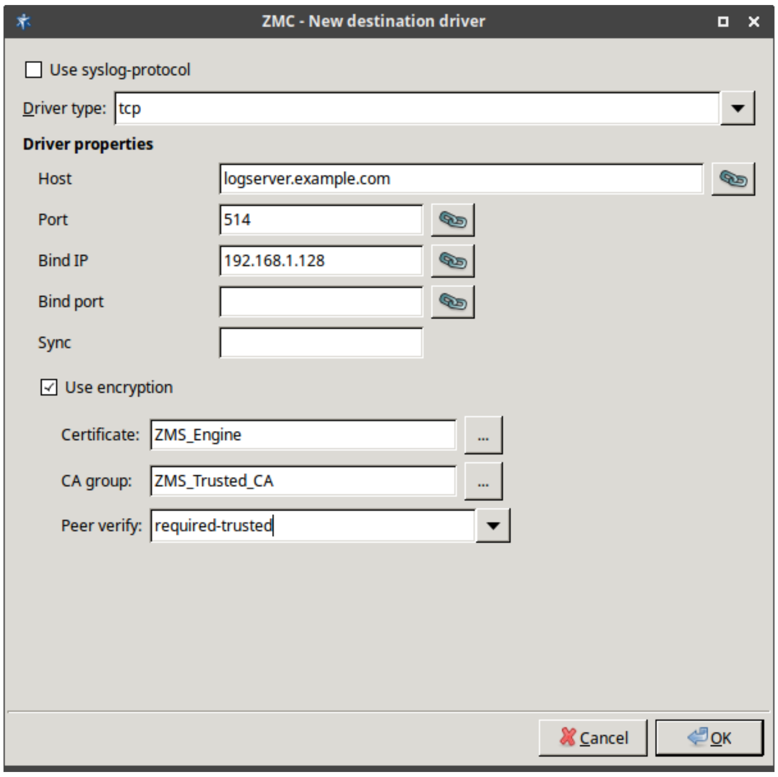
Name	Destination drivers
syslog	file

Drivers
file

```
file("/var/log/messages" template("$ISODATE $HOST $MSGHDR$MSG
\n") template_escape(no) fsync(no) create_dirs(no) flags())
```

Figure 7.16. Creating a new syslog destination

Step 2. Select **Drivers** > **New**, then **Driver type** > **tcp**.



ZMC - New destination driver

☐ Use syslog-protocol

Driver type: tcp

Driver properties

Host	logserver.example.com
Port	514
Bind IP	192.168.1.128
Bind port	
Sync	

☑ Use encryption

Certificate:	ZMS_Engine
CA group:	ZMS_Trusted_CA
Peer verify:	required-trusted

Figure 7.17. Configuring the syslog destination

Step 3. Set the **Use syslog-protocol** option to enabled, if you want the messages to be formatted according to the new IETF syslog protocol standard (RFC5424).

Step 4. Set the hostname and the port of your logserver in the **Host** and **Port** fields.

Step 5. Select the network interface of Zorp that faces the logserver from the **Bind IP** field.

Step 6. Select **Use encryption**.

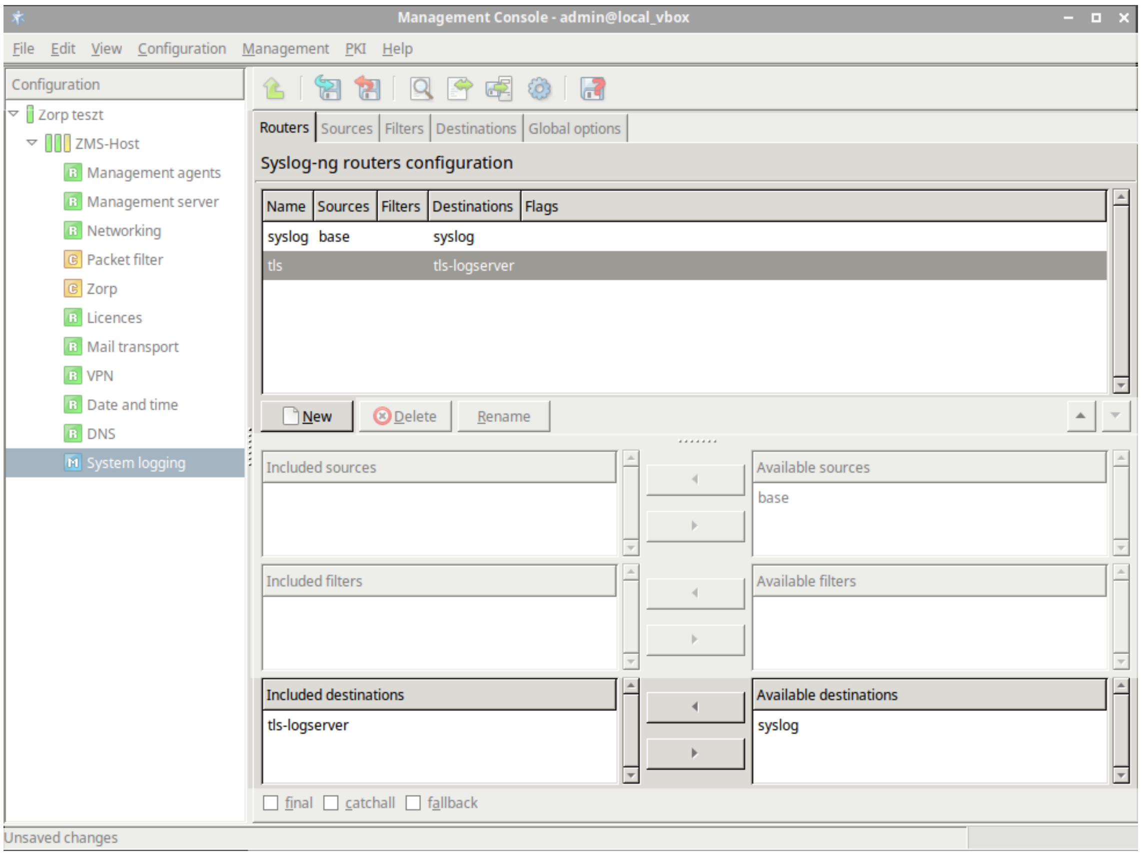
Step 7. If your logserver requires mutual authentication, that is, it checks the certificates of the log clients, select the certificate Zorp should show to the logserver from the **Certificate** field.

Step 8. Select the trusted CA group that contains the certificate of the CA that signed the certificate of the logserver from the **CA Group** field.

Step 9. By default, Zorp will verify the certificate of the logserver, and accept only a valid certificate. It is possible to have less strict criteria by modifying the **Peer verify** option. For details on the possible values, see *Section 3.2.5, Certificate verification options* in *Zorp Professional 7 Reference Guide*.

Step 10. Click **OK**.

Step 11. Select the **Router** tab, add a new router and name it, for example, to TLS.



Syslog-ng routers configuration

Name	Sources	Filters	Destinations	Flags
syslog	base		syslog	
tls			tls-logserver	

Included sources	Available sources
	base

Included filters	Available filters

Included destinations	Available destinations
tls-logserver	syslog

Figure 7.18. Configuring the syslog router

Step 12. Add the earlier defined new destination to this router.

Chapter 8. The Text editor plugin

All the essential parts of Zorp configuration have a corresponding custom configuration component in ZMC. In general, auxiliary services that are not part of the default Zorp configuration but installed separately, for example, a Snort Intrusion Detection System (IDS) service, ZMC provides a generic tool, the Text editor plugin configuration component to edit text-based configuration files. Since practically all Unix services work with text-based configuration files, the **Text editor** plugin can be used to configure all of them remotely.

Although the Text editor plugin provides only a simple text editor functionality, if you use this tool to configure the custom services, they are placed under the control of ZMS. From ZMC, the configuration changes are first committed to the ZMS server and stored there. After an upload command, these configurations are handled by the zms-transfer-agent on the corresponding Zorp firewall. So any service installed on the Zorp host are, in effect, ZMS-managed.

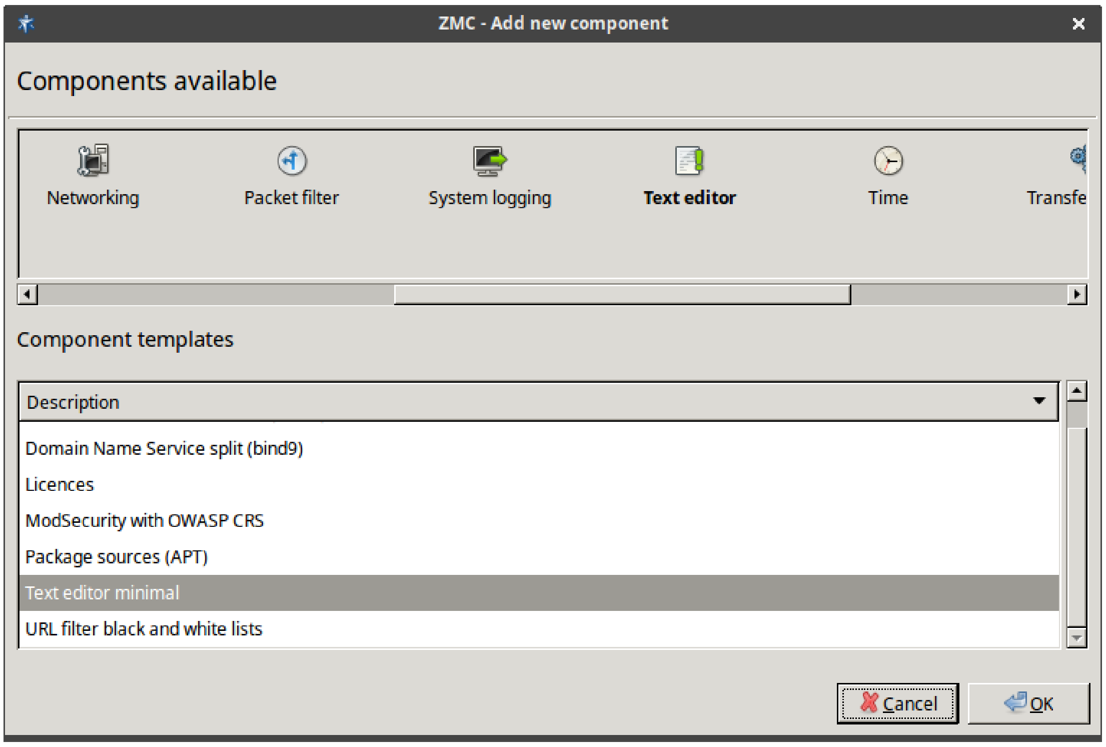
The number of **Text editor** plugins is not limited, you can add as many of them as needed. Furthermore, a single plugin can be used to edit more than a single text file.

8.1. Using the Text editor plugin

The Text editor plugin in ZMC is called `Text Editor`. It can be added to the list of configuration components the usual way.

8.1.1. Procedure – Configure services with the Text editor plugin

Step 1. Select a configuration template.
By default, there are some predefined configuration templates available, these are actually configuration file skeletons for the given services.

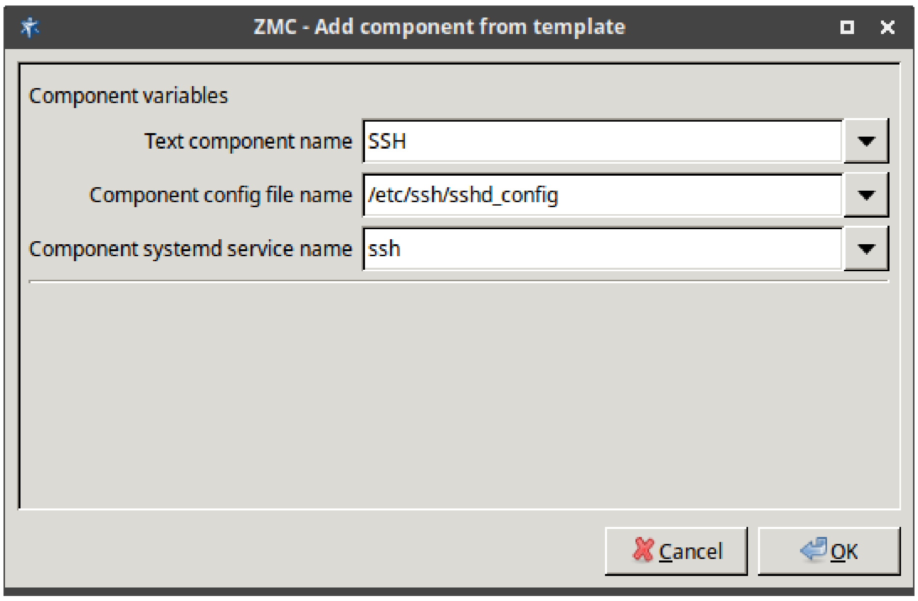


Domain Name Service split (bind9)
Licences
ModSecurity with OWASP CRS
Package sources (APT)
Text editor minimal
URL filter black and white lists

Figure 8.1. Default templates for the Text editor plugin

Unless you want to configure DNS or NTP services, or other predefined text components, select the *Text editor minimal* template.

The following configuration window appears.



Text component name: SSH
Component config file name: /etc/ssh/sshd_config
Component systemd service name: ssh

Figure 8.2. Configuration window for the Text editor plugin

Step 2. Provide the name of the *Text component*.

This parameter sets a label for the component.

Step 3. Provide the name of the *Component configuration file*.
Component configuration file name refers to the actual configuration file on the host that you want to edit.

It is most likely a file in `/etc` or in one of its subdirectories since this is the default location for configuration files.

Step 4. Set the *Component systemd service name*.
Component systemd service name refers to the name of the systemd service that is used to start, stop, restart, reload or check the status of a given service. Without providing this option it is not possible control the service from ZMC.

Note
ZMC runs the `/bin/systemctl start/stop/restart/reload/status` command with the given parameter.

Step 5. Click **OK** after setting the necessary parameters.
The main window of ZMC reappears with an empty component pane. This is normal. Before working with the desired configuration file it is possible to download the content from the selected host.

Step 6. Download and edit content from the selected host.
Use the following button in the button bar to download the file to ZMC:

You can edit the configuration file.

Step 7. Propagate the modifications to the firewall.
The steps are identical to that of the other components: commit the edited file to ZMS and then upload it to the host. Finally, the service can be started/stopped/restarted or reloaded the usual way.

Besides the file download option, the Text editor plugin provides some unique features compared to the other components.

8.1.2. Procedure – Use the additional features of Text editor plugin

Step 1. To simplify working with configuration files, you can insert precreated configuration file segments with the help of the **Insert file** button on the button bar.

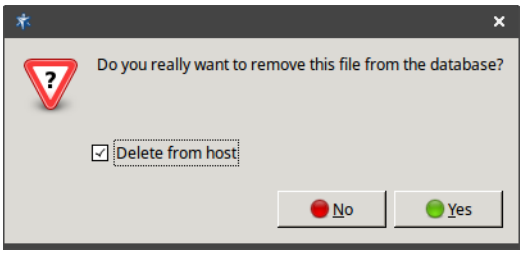
Note
The file to be inserted must be present on the ZMC machine. If you insert more than one files, they appear concatenated in the main workspace.

Step 2. You can create new files with the **New** button on the button bar.

If you create more than one file, they appear on different tabs in the main workspace of the **Text editor** plugin. This way a single plugin can host several different files.

Note
If you create a file with the Text Editor component, ZMC sets the permissions of the file and the directory to `root:root 700`, regardless of the original permissions or the umask settings of the system.

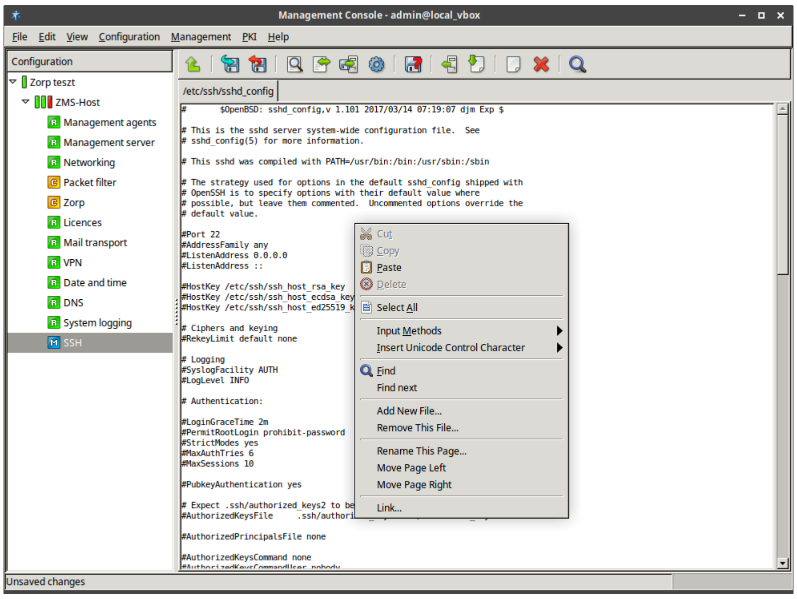
Step 3. You can remove configuration files with the **Remove** button.
By default, the **Remove** button only removes the configuration file from the ZMS database. If you want to remove the file from the host as well, check the appropriate checkbox in the **Remove** dialog box.


Do you really want to remove this file from the database?

☑ Delete from host

Figure 8.3. The Remove file dialog box

Step 4. You can also place links to linkable objects of the configuration in a file, if you right-click in the main workspace of the **Text editor** plugin and select **Link... from** the local menu.



```
#       $OpenBSD: sshd_config,v 1.101 2017/03/14 07:19:07 djm Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_k

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be
#AuthorizedKeysFile     .ssh/authori

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
```

*Figure 8.4. The local menu of the **Text editor** plugin*

Note
There is no restriction on what file can be administered via the **Text editor** plugin except that it must be a text-based file.

Chapter 9. Native services

The default Zorp installation includes some service components that are typically useful in networking environments. These are BIND for DNS traffic, NTP, and Postfix for SMTP traffic. The use of these services is not mandatory, however, they can help solving three particularly problematic issues of network configuration. Once configured, access to these services (so called local services), as well as remote SSH access to the Zorp host must be enabled separately by adding a local service. See *Section 9.4, Local services on Zorp (p. 232)* for details.

- *Name resolution*
- *Time synchronization*
- *Secure e-mail services*
- *Local Zorp services*

For enhanced security, some of these services run in a chrooted environment, otherwise known as a jail. A jail is a special, limited directory structure where the service executables and all the accompanying files, such as configuration files, are installed. The service that is jailed can only access this limited part of the file system hierarchy and is unaware of the rest of the file system. The 'chrooted environment' is a virtual subtree of the full file system, and the top of this subtree is seen by the chrooted (jailed) service as the root `'/'` directory. From the service's point of view, the jail is a complete file system in the sense that it contains all directories the service needs access to. For example, if the service needs a library from `/lib` or from `/usr/lib`, these two directories together with the actual `lib` files needed are included (copied) in the jail environment. The jail isolates the service from the rest of the system, so even if its security is compromised, it can only destruct the jail and cannot affect the rest of the system.

Note
Starting with Zorp 3.4, the BIND and NTP services do not run in jail, but use AppArmor instead. It is possible to set the BIND service to run in jail automatically.

9.1. BIND

BIND is the industry standard DNS solution used in the vast majority of Linux/Unix based name resolution services. It has three different “branches”, ISC BIND 4, 8 and 9, the most advanced development taking place in the ISC BIND 9 branch.

Installing Zorp automatically installs an ISC BIND 9 version. BIND shipped with Zorp is a full implementation of the most up-to-date version of the 9 branch, so theoretically it is possible to configure it for any DNS server role. It is hosted on a firewall, however, such liberal use of BIND is not recommended. Instead, it should rather be used as a limited-purpose DNS server. The following two examples are such possible configurations.

9.1.1. BIND operation modes

Example 9.1. Forward-only DNS server
In this scenario, BIND does not store zone information of any kind, instead, it simply forwards all name resolution requests to a designated nameserver located elsewhere. This way, BIND configuration and maintenance is minimal while name resolution traffic is optimized: BIND caches resolved name-to-IP address mappings, thereby saving some bandwidth and improving name resolution speed.

This setup is especially recommended for small to medium-sized networks where DNS zone information of the company is maintained off-site, typically at an ISP, and thus maintaining a dedicated nameserver only for Internet name resolution is not economical.

In this setup BIND operates essentially as a DNS proxy.

Example 9.2. Split-DNS implementation
In this setup two sets of records on the DNS server are maintained:

- a public set which is available for general access, and
- a private set that is available for internal users only.

With this setup it is possible for a company to both maintain its own public DNS zone records (SOA, NS, MX and A records for hosts running popular services like WWW or FTP) and some internal DNS records for servers that are (and must be) available for internal users only.

This setup is recommended for companies wishing to host their own DNS zone database but the number of external name resolution requests does not facilitate the use of a dedicated DNS server.

9.1.2. Configuring BIND with ZMC

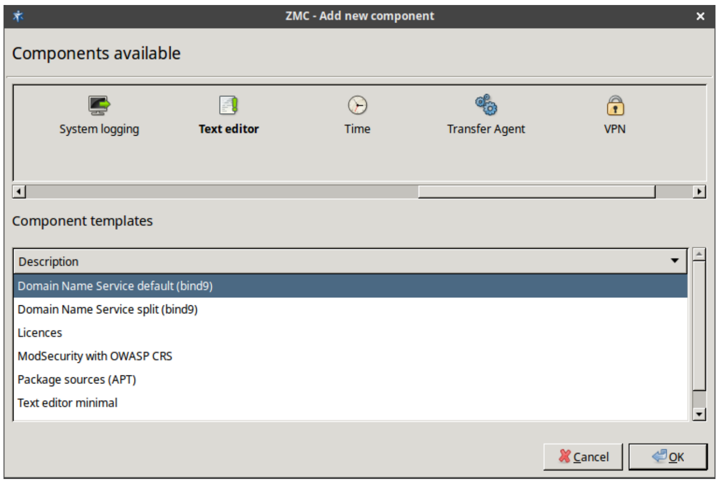
The configuration of BIND is stored under `/etc/bind/`, where the most important file is `named.conf`. This is the general configuration file. Zone database information is stored in `db.domainname` files under the same directory.

ZMC does not offer a dedicated component to edit BIND configuration, instead, the `Text editor` component offers preconfigured templates for this task.

9.1.2.1. Procedure – Configuring BIND with ZMC

Step 1. Add the `Text editor` as a new component.

Step 2. Select a template to be used with the Text Editor.



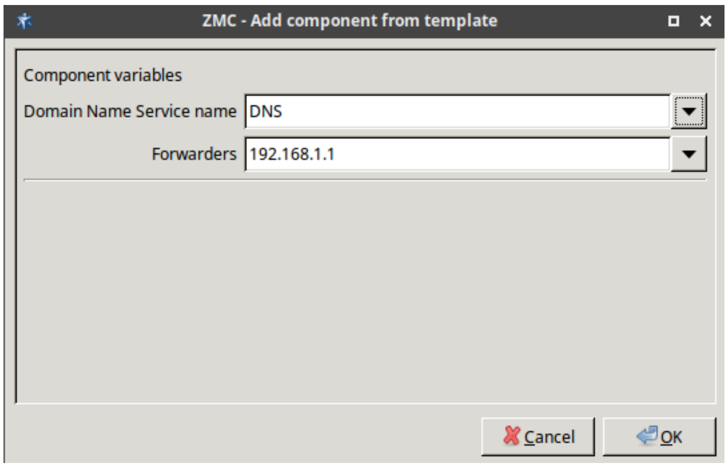
Domain Name Service default (bind9)
Domain Name Service split (bind9)
Licences
ModSecurity with OWASP CRS
Package sources (APT)
Text editor minimal

Figure 9.1. Selecting a Text editor template

Select one of the first two templates depending on whether your want a split DNS configuration or not.

Click **OK**.

Step 3. Configure the basic settings in the opening window.



Domain Name Service name: DNS

Forwarders: 192.168.1.1

Figure 9.2. Configuring basic BIND settings

Step a. Provide the **Domain Name Service** name.
This parameter simply specifies a label for the component that appears in the components pane.

Step b. Specify **Query source**.
This parameter defines where the outgoing name resolution requests originate on the firewall.

Note
Prior to BIND 8.1 the source port was 53 (just like the destination port), but since then BIND uses a port from the dynamic range, 5300 by default.

This might be important in back-to-back firewall configurations where there is another firewall in front of this instance of Zorp. To allow outgoing DNS requests, the front firewall must know the source port used by the BIND service.

Besides supplying an alternate port number, you can supply a fixed IP address of Zorp if it has more than one in the required direction. If this setting is not relevant in your network environment, choose the IP address of the outside interface.

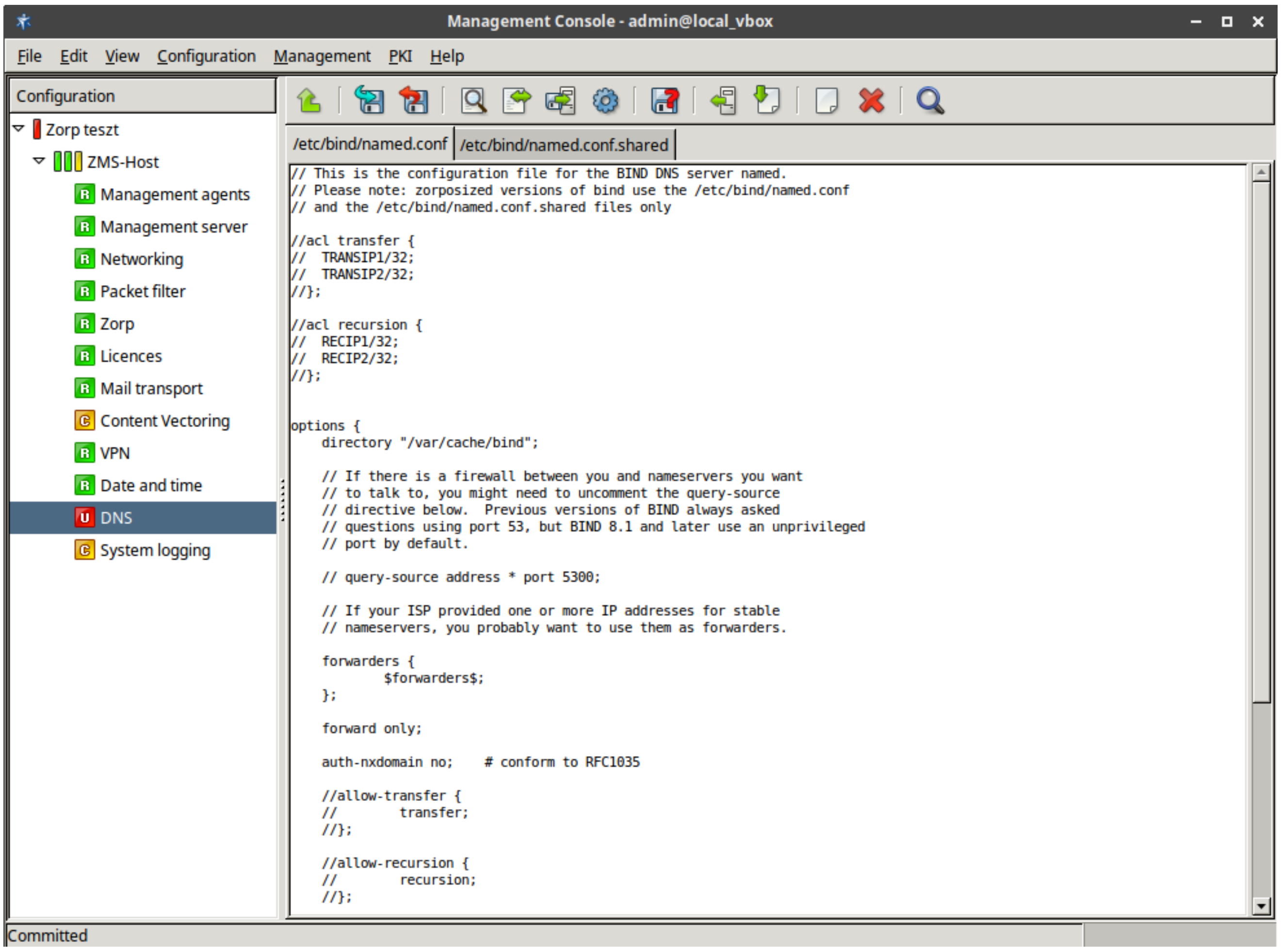
Step c. Define **Forwarders**.
In a Zorp installation, BIND is usually configured as a forward-only nameserver. If you configure a forwarder, BIND does not resolve names recursively on the Internet, but instead it forwards all name resolution requests to the DNS server specified as the forwarder.

After entering values for these parameters the first round of BIND configuration is ready, a functional forward-only nameserver is in place.

Step 4. To permit access to the BIND service, enable the *`dns`* local service. If you plan to host zone database information on the Zorp Gateway, enable the *`dns-zonetrans`* local service as well. See *Section 9.4, Local services on Zorp (p. 232)* for details.

Note
If you use zone transfer, be careful with selecting which zones you accept zone transfer requests from.

If you go back to the `DNS configuration` component you can see the structure of the `named.conf` file with the values entered for query-source and forwarders.



```
// This is the configuration file for the BIND DNS server named.
// Please note: zorposized versions of bind use the /etc/bind/named.conf
// and the /etc/bind/named.conf.shared files only

//acl transfer {
//  TRANSIP1/32;
//  TRANSIP2/32;
//};

//acl recursion {
//  RECIP1/32;
//  RECIP2/32;
//};


options {
    directory "/var/cache/bind";

    // If there is a firewall between you and nameservers you want
    // to talk to, you might need to uncomment the query-source
    // directive below.  Previous versions of BIND always asked
    // questions using port 53, but BIND 8.1 and later use an unprivileged
    // port by default.

    // query-source address * port 5300;

    // If your ISP provided one or more IP addresses for stable
    // nameservers, you probably want to use them as forwarders.

    forwarders {
            $forwarders$;
    };

    forward only;

    auth-nxdomain no;    # conform to RFC1035

    //allow-transfer {
    //        transfer;
    //};

    //allow-recursion {
    //        recursion;
    //};
```

Figure 9.3. `named.conf` in ZMC

In addition, this tool is suitable for editing the `named.conf` file only. If you want to host zone database files (`db.domainname`) on the firewall, you can edit the files separately if you create a new file in the `Text Editor` component. However, for forward-only nameserver configurations editing the `named.conf` file is generally sufficient.

9.1.3. Procedure – Setting up split-DNS configuration

Setting up a split DNS service is useful in networks where both external and internal name resolution is performed using the same DNS server – in this case the Zorp firewall. Since hiding the internal namespace from external visitors is a basic security requirement, you have to set up the DNS service in a way that it does not resolve internal names for external resolvers. In other words, for all DNS zones stored on the server you have to specify which networks can query for records in the given zone.

Step 1. Add the Text Editor component.

Step 2. Select the **split-dns** template.
Two skeleton files are created, a `named.conf` and a `named.conf.shared`.

The `named.conf.shared` file holds records and configuration settings that are shared between external and internal name resolution operations, while `named.conf` has options to specify internal and external networks (`internalips` and `externalips`). These networks can then be referenced in `db.domainname` file(s) to specify which networks can have access to what records.

For more information on split-dns configuration and DNS configuration in general, see *Appendix C, Further readings (p. 485)*.

9.2. NTP

Accurate timekeeping is very important for firewalls. Without reliable time data, security log analysis is very difficult and can lead to false results. Besides, if the firewall provides any auxiliary services that use timestamping, for example, a mail service, false time values can be disturbing, too. One of the few things that are still operating in the original, free and liberal spirit of the Internet is the Network Time Protocol service. There are a number of timeservers on the Internet that allow free connections from anywhere. For a complete list, see *Appendix C, Further readings (p. 485)*.

Companies typically connect to a Stratus 2-level timeserver on the Internet and then distribute time within their organization from a single time server source. Using the native NTP service, Zorp can function as a central timeserver for the entire organization, if needed. This service generally does not put a heavy load on the machine, nor does it pose a significant security risk, so it is generally acceptable to use Zorp as a timeserver for the internal machines.

Unlike application proxy plugins, native services operate as feature-complete software components, so the NTP component in Zorp is a real NTP server. NTP is generally not suitable for proxying, since the latency of the proxy component would not be constant but load–dependent rather. Packet filtering could work for NTP but application-level handling of traffic generally offers a higher level of security. NTP is handled by a native service based on these reasons.

9.2.1. Procedure – Configuring NTP with ZMC

NTP has a separate configuration component in ZMC.

Step 1. Add the `Time` component to the Zorp host in ZMC. It is recommended to select the *`Date and time`* template.

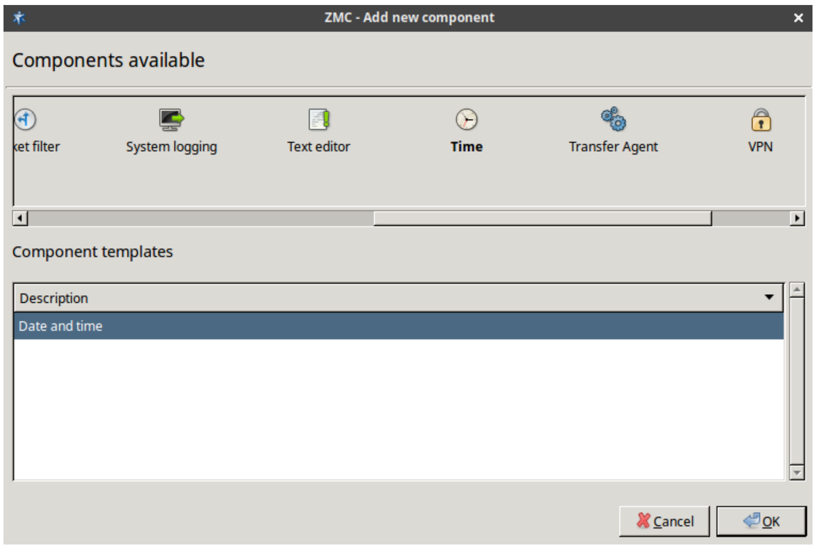
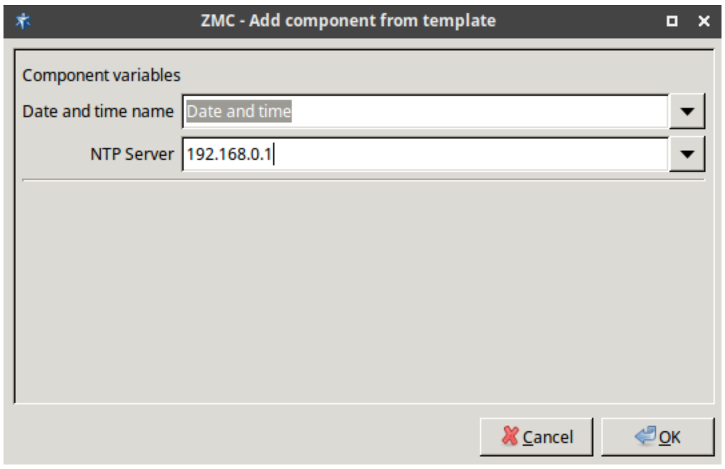


Figure 9.4. Adding the Time component

Step 2. Name the component.

Step 3. Specify a time server with which Zorp synchronizes its system time.

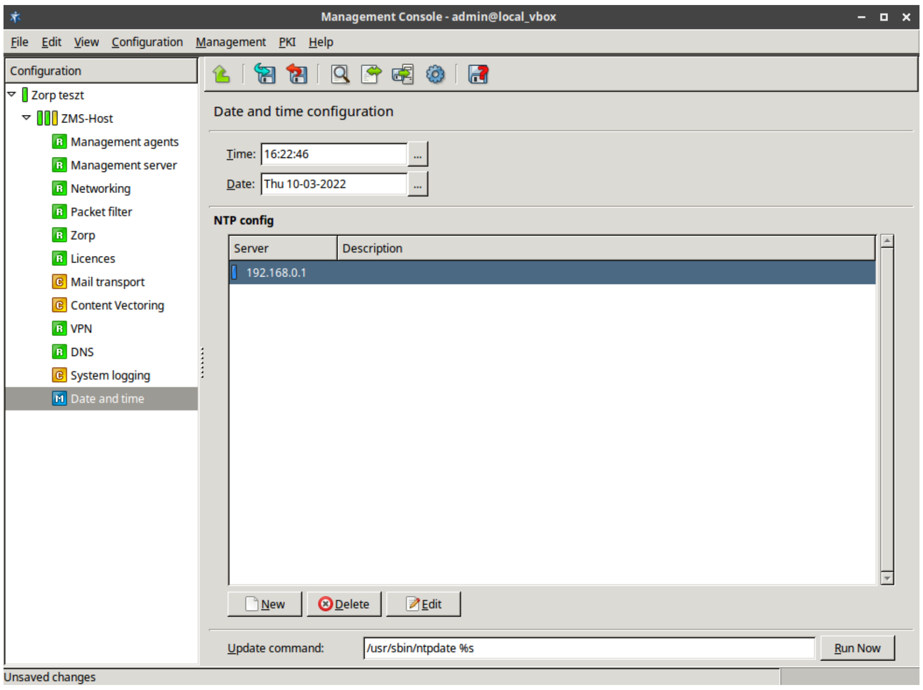


Date and time name: Date and time

NTP Server: 192.168.0.1

Figure 9.5. Selecting a time server to synchronize with

Step 4. Configure the **Time** subcomponents. Date and time can be updated from the specified time server by clicking **Run now**, or set manually.



Date and time configuration

Time: 16:22:46

Date: Thu 10-03-2022

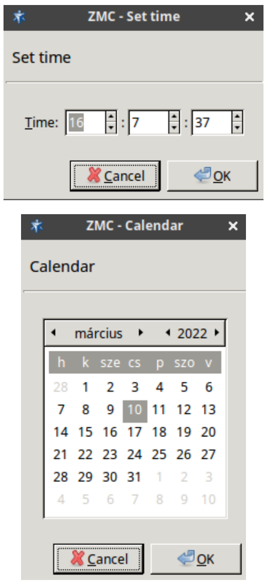
NTP config

Server	Description
192.168.0.1	

Update command: /usr/sbin/ntpdate %s

Figure 9.6. The Time component

Step a. Set date and time information manually, using the three-dotted (...) button.



h	k	sze	cs	p	szo	v
28	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Figure 9.7. Editing time and date values manually

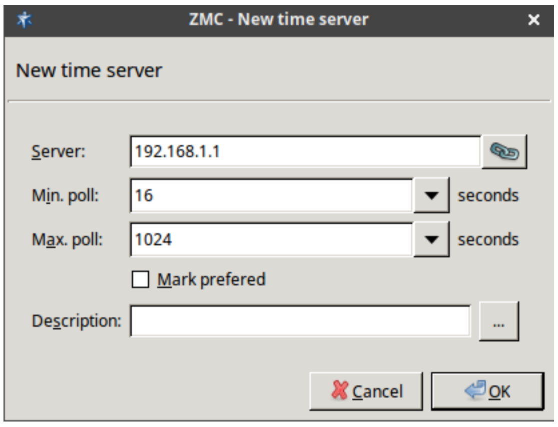
If the date and time values are set manually, a dialog box gives a warning that instead of setting these values manually, it is recommended to force an update from the configured timeserver with the `ntpdate` command. To synchronize the time to the time server, click the **Run now** button in the bottom of the window.

It is recommended to use `ntpdate` instead of manually tuning date and time values because the Internet time is probably more accurate than other, local time sources.

Step b. If you want to permit your clients to synchronize their clocks to the Zorp host, enable the *`ntp`* local service. See *Section 9.4, Local services on Zorp (p. 232)* for details.

Step c. List the NTP servers Zorp can communicate with.

For time synchronization fault tolerance, It is recommended to add at least two servers to the list. Click **New** and enter the address of the NTP server, as well as the interval when the clock of the Zorp host should be synchronized to the NTP server.



Server: 192.168.1.1
Min. poll: 16 seconds
Max. poll: 1024 seconds
Mark prefered
Description:

Figure 9.8. Adding a new NTP server

9.2.2. Status and statistics

The status of the configured time servers is indicated by leds of different colors before the name of the server. Hovering the mouse over a server displays the following statistics about the connection in a tooltip (for details on these values see *Appendix C, Further readings (p. 485)*):

- *Tally code*:
- *Ref. ID*: It is the reference ID (0.0.0.0 if unknown).
- *Stratum*: It is the place of the server in the 'clock strata' hierarchy. Stratum 1 systems are synchronised to an accurate external clock; stratum 2 systems derive their time from one or more stratum 1 systems, and so on.
- *Type*: It is the type of the peer (local, unicast, multicast or broadcast).
- *Last arrived*: It is the time value when the last packet was received.
- *Polling interval*: It defines the period between two queries in seconds.
- *Reachablility*: It is a register indicating the reachability of the peer. A peer is considered to be reachable if at least one bit in this register is set to one.
- *Delay*: It is the current estimated delay in milliseconds.
- *Offset*: It is the current estimated offset in milliseconds.

- *Jitter*: It is the estimated time error of the peer clock (measured as an exponential average of RMS time differences).
- *Status*: The status displays the syncronisation status of the NTP server.

9.3. Postfix

SMTP mail handling in Zorp is very flexible and is designed to allow for as many different e-mail “needs” as possible. Based on the size, profile and security requirements of a company, there are a number of configurations possible for handling email traffic.

Very small companies trust their ISP to host SMTP service for them and only connect with a mail retrieval (post office) protocol (POP3 or IMAP) to download the mail and use the ISP's mail server as their outgoing SMTP server. Larger companies may have their own SMTP server but still use the ISP's mail server as their official mail exchanger and only relay mail between the two. Companies that need maximal protection, have a fully functional, DNS-registered mailserver. The next level of security for companies can be achieved by sophisticated mail routing architecture, multiple domains and complex email traffic rules.

Zorp aims to provide protection support for all types of SMTP requirements. It has a proxy class for SMTP that is the primary tool for handling SMTP traffic. It is not a fully functional mail server but a fully transparent filter module rather. It does not send and receive SMTP mail messages and it does not have a local mail store either. This proxy can interoperate with antivirus software for filtering viruses in SMTP traffic. With the SmtpProxy or a customized, derived version of it most SMTP firewalling needs can be fulfilled.

There are, however, cases when simply proxying SMTP traffic is not enough and some more intelligent mail handling procedure is required due to the organization's special needs.

Example 9.3. Special requirements on mail handling

1. If a company maintains multiple mail domains and/or complex mail routing rules are needed using transport tables.
2. If a company aims to avoid time-outs when antivirus filtering is enabled and large attachments need to be scanned. SmtpProxy will only accept (acknowledge) a mail message after it has arrived and has been scanned for viruses unlike most MTAs, which may lead to timeout situations when communicating with other, real MTAs on the Internet.

For such cases Zorp installs a fully functional Postfix service besides the SmtpProxy. It is fully functional and virtually, any setups and configurations possible with a Postfix mail server, are also possible here. It does not mean that Zorp shall be operated as a generic mail server for users, however, sophisticated SMTP configurations are possible with it.

Note
By default, Zorp does not install a mailbox protocol server program, because a firewall should not run a POP3 or IMAP server.

The Postfix component can also provide SMTP delivery service for local services, and similarly to syslog-ng and other services, it has to be able to send e-mails. The local delivery of e-mails, however, shall not be allowed, if possible.

Note
The Postfix native service is not intended to replace the SmtpProxy application proxy in SMTP–handling configurations.

Even if the configuration options of SmtpProxy are not adequate, it is still recommended for the SMTP mail service handling to be 'front-end' at the firewall, which, after proxy-level filtering, passes SMTP traffic to the Postfix service.

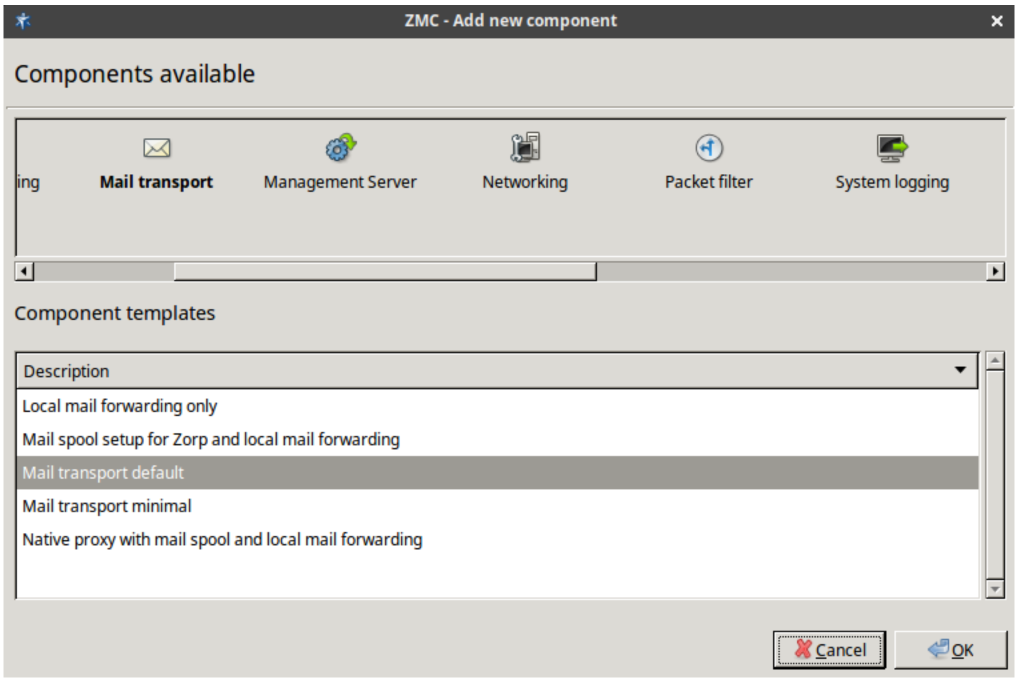
As the possible uses of the Postfix component are so versatile, it is not possible to cover even the most typical ones in this chapter. Nor is it a firewall administrator's task to set up a complex mail routing architecture. Therefore only a brief introduction of the configuration interface is presented. For more information and details on Postfix, see *Appendix C, Further readings (p. 485)*.

9.3.1. Configuring Postfix with ZMC

You can accomplish Postfix configuration through a set of configuration files represented by the **Mail transport** component. This plugin has five tabs, corresponding to configuration files in `/etc/postfix` on the firewall.

9.3.1.1. Procedure – Configuring Postfix with ZMC

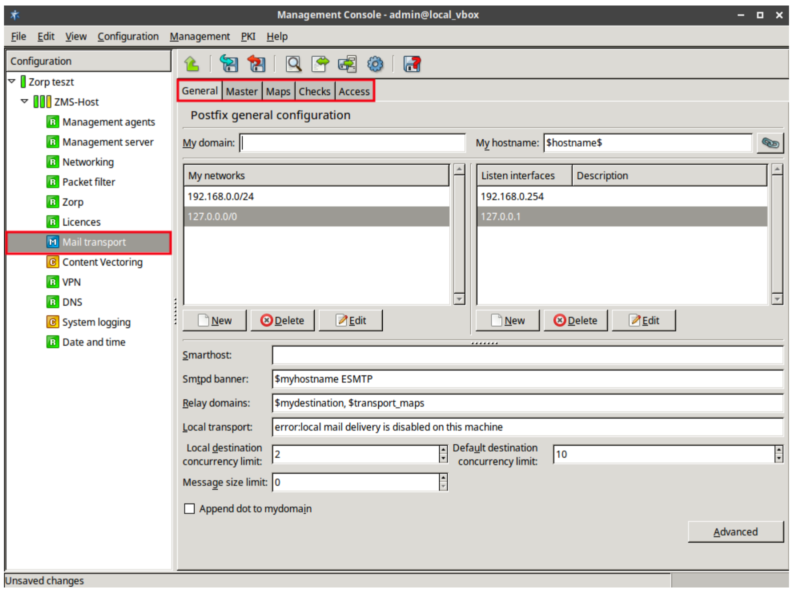
Step 1. Add the `Mail transport` component to the Zorp host in ZMC. Select a template suitable for your needs, for example, the *`Mail transport default`* template.



Description
Local mail forwarding only
Mail spool setup for Zorp and local mail forwarding
Mail transport default
Mail transport minimal
Native proxy with mail spool and local mail forwarding

Figure 9.9. Adding the Mail transport component

Step 2. Open the configuration tabs.



Postfix general configuration

My domain:

My hostname: $hostname$

My networks
192.168.0.0/24
127.0.0.0/0

Listen interfaces	Description
192.168.0.254	
127.0.0.1	

Smarthost:	
Smtpd banner:	$myhostname ESMTP
Relay domains:	$mydestination, $transport_maps
Local transport:	error:local mail delivery is disabled on this machine
Local destination concurrency limit:	2
Default destination concurrency limit:	10
Message size limit:	0

☐ Append dot to mydomain

Figure 9.10. Configuration tabs in the Mail transport plugin

Step 3. Specify parameters in the **General** tab.

Step a. Provide **My domain**.

It specifies the DNS domain of Zorp which, in turn, defines what domain it receives mail for. Receiving mail for other domains is also possible. For details, see *Appendix C, Further readings (p. 485)* for a reference on mail administration.

Step b. Enter **My Hostname**.

It is the name of Zorp, exactly as it is registered in DNS. The MX record in DNS must point to this name, so it is important to specify it correctly.

Step c. Provide **My networks**.

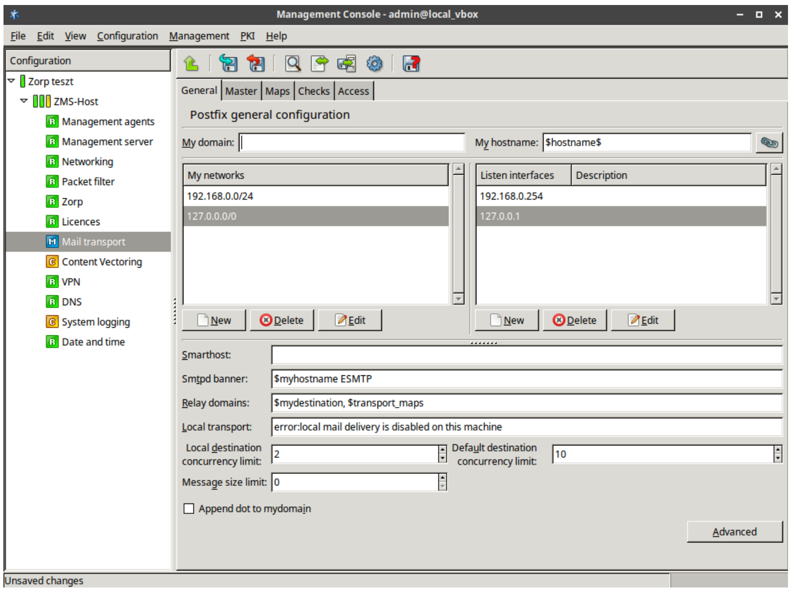
It specifies what IP networks Postfix accepts outgoing mail from, in other words, for which networks it acts as a mail relay.

Note

Unless explicitly required by your networking requirements, do not to list all your internal networks. It can result in all your hosts being able to send mails individually and directly, which might not be optimal from security point of view. For example, viruses usually contain an SMTP component for sending mail that should not be let through the firewall.

If you only have a single mail server for handling external SMTP messages, list the mail server's single IP address. Correspondingly, list only those network interfaces of Zorp as Listen interfaces, on which you want to handle incoming mail traffic.

The rest of the parameters on the **General** tab are more special settings and their use depends on the configuration needs.



Postfix general configuration

My domain:

My hostname: $hostname$

My networks
192.168.0.0/24
127.0.0.0/0

Listen interfaces	Description
192.168.0.254	
127.0.0.1	

Smarthost:

Smtpd banner: $myhostname ESMTP

Relay domains: $mydestination, $transport_maps

Local transport: error:local mail delivery is disabled on this machine

Local destination concurrency limit: 2

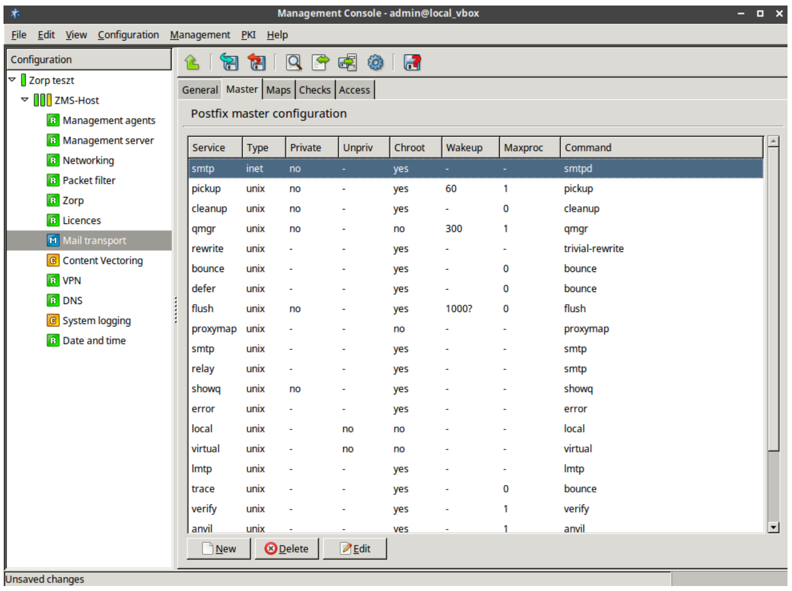
Default destination concurrency limit: 10

Message size limit: 0

☐ Append dot to mydomain

Figure 9.11. Essential components of Postfix configuration

Step 4. Configure settings on the **Master** tab.



Postfix master configuration

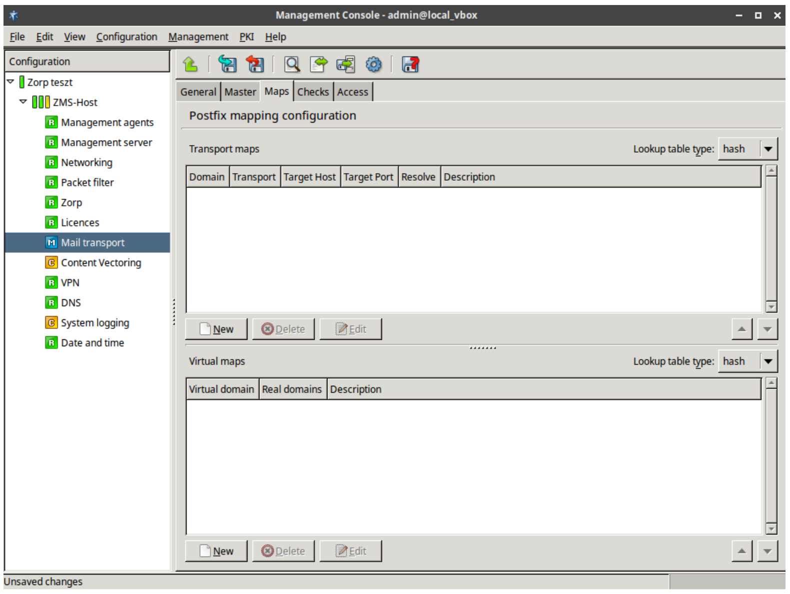
Service	Type	Private	Unpriv	Chroot	Wakeup	Maxproc	Command
smtp	inet	no	-	yes	-	-	smtpd
pickup	unix	no	-	yes	60	1	pickup
cleanup	unix	no	-	yes	-	0	cleanup
qmgr	unix	no	-	no	300	1	qmgr
rewrite	unix	-	-	yes	-	-	trivial-rewrite
bounce	unix	-	-	yes	-	0	bounce
defer	unix	-	-	yes	-	0	bounce
flush	unix	no	-	yes	1000?	0	flush
proxymap	unix	-	-	no	-	-	proxymap
smtp	unix	-	-	yes	-	-	smtp
relay	unix	-	-	yes	-	-	smtp
showq	unix	no	-	yes	-	-	showq
error	unix	-	-	yes	-	-	error
local	unix	-	no	no	-	-	local
virtual	unix	-	no	no	-	-	virtual
lmtp	unix	-	-	yes	-	-	lmtp
trace	unix	-	-	yes	-	0	bounce
verify	unix	-	-	yes	-	1	verify
anvil	unix	-	-	yes	-	1	anvil

Figure 9.12. The Master tab

Configure the settings if you have a Mail Scanner or Amavisd-new–based antivirus solution.

The **Master** tab of the Mail transport component corresponds to the `/etc/postfix/master.cf` file.

Step 5. Configure settings on the **Maps** tab to add transport and virtual maps to Postfix.


Management Console - admin@local_vbox
File Edit View Configuration Management PKI Help
Configuration
Zorp teszt
ZMS-Host
Management agents
Management server
Networking
Packet filter
Zorp
Licences
Mail transport
Content Vectoring
VPN
DNS
System logging
Date and time
General Master Maps Checks Access
Postfix mapping configuration
Transport maps
Domain Transport Target Host Target Port Resolve Description
New Delete Edit
Virtual maps
Virtual domain Real domains Description
New Delete Edit
Unsaved changes

Lookup table type: hash

Lookup table type: hash

Figure 9.13. The Maps tab

In order to route incoming mail from Zorp to different, internal mail domains, an SMTP transport map can be provided, with the IP address of the real, internal mail servers serving the given mail domains.

Step 6. Configure the **Checks** tab.

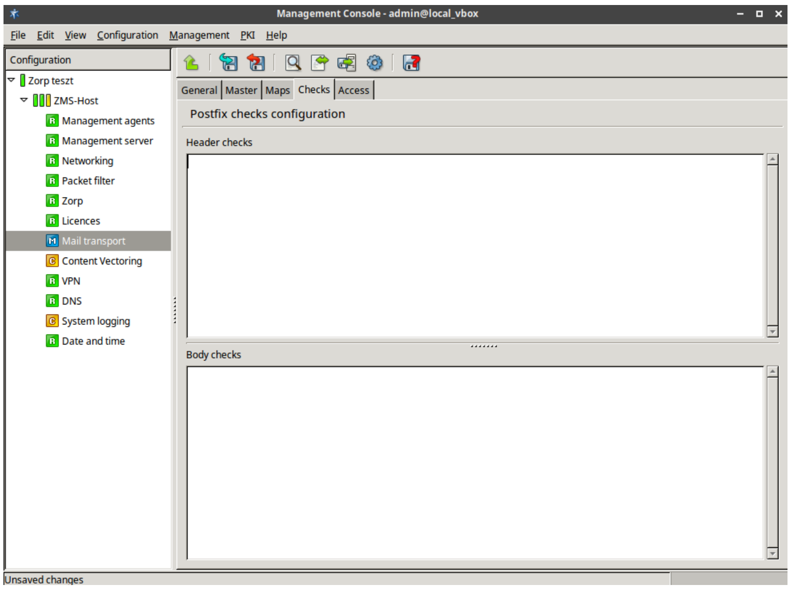


Figure 9.14. The Checks tab

This tab covers two Postfix configuration files, `/etc/postfix/header_checks` and `/etc/postfix/body_checks`. The method of the address checking can be either hash or regular expression (regexp). This can be selected from the **Lookup table type** combobox.

Step 7. Configure the **Access** tab.

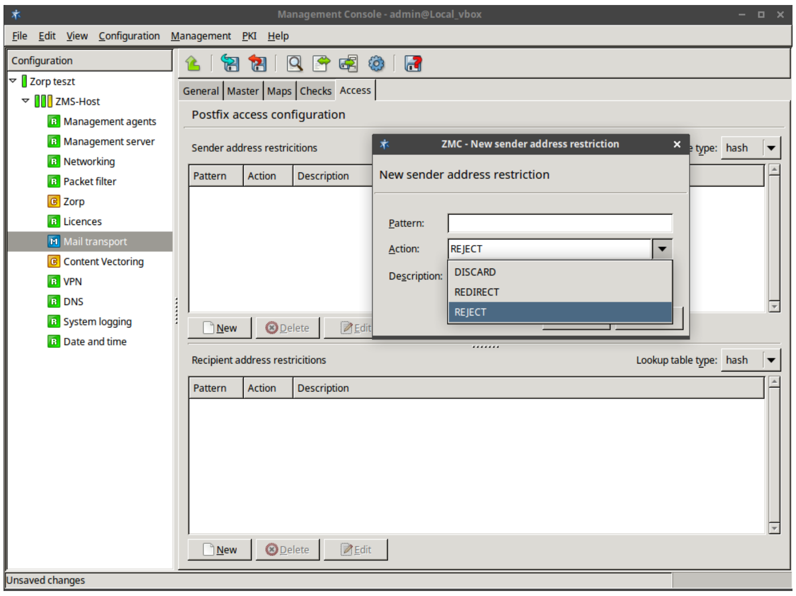

Management Console - admin@Local_vbox
File Edit View Configuration Management PKI Help
Configuration
Zorp teszt
ZMS-Host
Management agents
Management server
Networking
Packet filter
Zorp
Licences
Mail transport
Content Vectoring
VPN
DNS
System logging
Date and time
General Master Maps Checks Access
Postfix access configuration
Sender address restricitions
Pattern Action Description
New Delete Edit
Recipient address restrictions
Lookup table type: hash
Pattern Action Description
New Delete Edit
Unsaved changes
ZMC - New sender address restriction
New sender address restriction
Pattern:
Action: REJECT
Description:
DISCARD
REDIRECT
REJECT

Figure 9.15. The Access tab

In parallel with **Checks**, this tab covers `/etc/postfix/recipient_access` and `/etc/postfix/sender_access`.

Step 8. To permit access to the Postfix service, enable the *`smtp`* local service. See *Section 9.4, Local services on Zorp (p. 232)* for details.

Note
Choose the zones that are allowed to access the Postfix service carefully.

Together these tabs, actually, the files they directly correspond to, make up the majority of typical Postfix configuration.

As you see, it is technically possible to create a full featured SMTP server on the Zorp machine, but it is definitely not recommended.

Note
It is recommended to use the SMTP proxy to perform mail proxying on your Zorp firewall. The Postfix component shall only be used, for example, in case of complex mail routing requirements.

9.4. Local services on Zorp

Local services run on the elements of the Zorp Gateway System: on Zorp, ZMS, and ZCV hosts. Zorp hosts can provide the following services locally:

Warning
Local services can be accessed only by using IPv4. IPv6 access for local services is currently not supported.

- *ssh*: It enables remote SSH access to the Zorp host. It opens port TCP/22.
- *smtp*: It enables the transport of SMTP (e-mail) traffic. This local service must be enabled if you want to use the native Postfix service of Zorp to handle e-mail transfer (see *Section 9.3, Postfix (p. 227)*). It opens port TCP/25.
- *nagios-nrpe-server*: It enables *nagios-nrpe-server* to query the Zorp. This local service must be enabled if you want to monitor your Zorp with Nagios (see *Procedure 17.3, Monitoring Zorp with Nagios (p. 460)*. It opens port TCP/5666.
- *munin-node*: It enables Munin to query the Zorp. This local service must be enabled if you want to monitor your Zorp with Munin (see *Procedure 17.1, Monitoring Zorp with Munin (p. 459)*. It opens port TCP/4949.
- *ntp*: It enables clients to synchronize their system clocks to the clock of the Zorp host using NTP. This local service must be enabled if you want to use the native NTP service of Zorp (see *Section 9.2, NTP (p. 223)*). It opens port UDP/123.
- *identreject*: If it is enabled, Zorp rejects every traffic arriving to the 113/TCP port.
- *dns*: It enables clients to use the Zorp host as a DNS server. This local service must be enabled if you want to use the native BIND9 service of Zorp (see *Section 9.1, BIND (p. 218)*). It opens port UDP/53.
- *dns-zonetrans*: It enables clients to use the Zorp host as a DNS server. This local service must be enabled if you want to use the native BIND9 service of Zorp and enable zone transfer (see *Section 9.1, BIND (p. 218)*). It opens port TCP/53.
- *zmsgui*: It enables administrators to connect to ZMS with ZMC, and manage the Zorp Gateway System. It opens port TCP/1314.
- *zmsengine*: It enables communication between ZMS and the Zorp hosts. This local service must be enabled if a host is managed from ZMS. It opens ports TCP/1311 and port TCP/1313.
- *zmsagent*: It enables communication between the Zorp hosts and ZMS. This local service must be enabled on the ZMS host. It opens ports TCP/1310 and TCP/1312.

Note
Zorp automatically enables the services required for the management of the host: `zmsagent` for Zorp hosts; `zmsgui` and `zmsagent` for ZMS hosts. It is recommended to allow SSH as well.

Local services can be managed on the **Services** tab of the **Packet filter** ZMC component. For every local service, the **Name**, the used **Port** or **ICMP type)**, the **Protocol** (TCP, UDP or ICMP), and the **Target** parameters are displayed. If the value for the **Target** parameter is `ACCEPT` for a local service, the service is permitted, if the vaue is `REJECT` it is denied. To enable access to a local service on a host, complete the following steps.

9.4.1. Procedure – Enabling access to local services

Step 1. Navigate to the **Services** tab of the **Packet filter** ZMC component of the host and click **New**.

Step 2. Select the service from the **Local service** combobox. The port number and the type of the protocol is set automatically. Modify them only if you have a special configuration.

Step 3. Select the zones permitted to access the service, then click **Ok**. The packet filter rule corresponding to the new service is automatically added to the ruleset.

Step 4. To activate the new service, do not forget to **Commit** and **Upload** the changes to the host, and to **Reload** the packet filter using the **Control service** icon.

Chapter 10. Local firewall administration

Zorp, in cooperation with the ZMS and ZMC software components, is designed to be fully configurable from the graphical user interface of ZMC. Though this graphical administration is definitely the preferred method of management, it is possible to manually accomplish all the management and configuration tasks using a simple, character–based terminal console connection. In addition, the console–based administration provides some useful tools for troubleshooting scenarios that are not available through ZMC.

Local firewall administration, in this sense, does not necessarily refer to administration that takes place physically at the firewall machine using its local console and keyboard, but it also refers to setups where the character terminal of the firewall is reached through a secure network connection using SSH. The described administration is local in the sense that the configuration files are directly manipulated on the firewall machine, and not through the ZMS database.

Note
ZMS reads the configuration files of the firewall host only once, when it is bootstrapped. For details, see *Chapter 4, Registering new hosts (p. 53)*. After that, configuration changes are only downloaded to the host with the help of the transfer agent and are not parsed again by ZMS. Therefore, if you make local changes to a configuration file which is otherwise managed by ZMS, your configuration changes are overwritten when you next issue an `Upload` command from ZMS.

Configuration files that are not managed by ZMS, for example custom installed services on the firewall for which you do not define a Text Editor plugin, are not affected by this rule.

10.1. Linux

The components of Zorp Gateway run on Ubuntu-based operating systems, currently on Ubuntu 18.04 LTS. Therefore, for local management, the tools and procedures available are more or less the same as for any other Linux installation. If you install the Zorp host using the installation media of Balasys, only the most essential tools and services are installed by default. Although it is technically possible to install almost any kind of Linux software on a Zorp host, it is not recommended because custom installed tools may have a negative effect on the system in terms of security and stability. In fact, all local administration tasks can be accomplished by using the software tools that are available by default.

Linux is a technically sophisticated operating system that allows full access to and total control over itself for system administrators. Therefore, it is vital to have the sufficient skills and experience before administering it locally in a production environment.

Note
Untrained or inexperienced use can render any Linux system inoperable quickly, therefore extreme care is required when performing local administration.

This chapter is by no means intended to be a Linux tutorial. If you are unfamiliar with the concepts of general administration of Linux, consult some form of documentation before proceeding. There is excellent documentation available for Linux, both in printed and online forms. To avoid scattering of resources and material, the Linux Documentation Project (LDP) which manages the documentation tasks of Linux, provides a central access point to a thematically organized set of Linux documentation. Visit LDP site (http://www.ldp.org) for more information. Although the primary language of documentation is English, a substantial part of the material is either translated or is in the process of translation to other languages. If you prefer traditional, paper–based documentation forms, books on Linux administration are also available from major publishers worldwide. For more information, see *Appendix C, Further readings (p. 485)*.

10.2. Login to the firewall

A basic rule of working with any operating system is that only tasks that require system administrator (root) rights must be performed using a system administrator's account. All other tasks must be performed as a normal, non-privileged user. This is especially true for security-sensitive environments, such as a firewall.

Therefore, for every administrator of the firewall who needs local access, a normal user account must be created. Even if a firewall is managed by a team of administrators, typically only senior–level staff must be provided local logon rights. To preserve accountability and to maintain an adequate level of security, a separate account must be created for each administrator. These accounts are normal user accounts with no special privileges. However, to perform administrative tasks, special, root level access is needed to the services involved in the administrative action.

Linux provides the `sudo` tool to grant root level access to dedicated normal users. `sudo` allows the users to run only certain commands (specified by the root user) as root.

The `sudo` utility provides a secure method of having root level access to parts of the system. It is especially useful if there is more than one user who is potentially local administrator. To use `sudo`, all users who need root access must be listed in the file `/etc/sudoers`. The `sudo` command then allows these users to run only certain commands (specified by the root user) as root. This selective method is more secure than providing full root level access for a user to all parts of the system. It also allows for a more granular control over user activities on the system. `sudo` has configurable options as well as default settings; more information on these can be found in its manual pages (*man sudo*).

Tip
You can access Linux/Unix manual pages easily from Windows environment as well. If you have a Mozilla or Firefox browser, type *man sudo* in the address line and the browser opens immediately the sudo project's homepage.

Local administration of the firewall can be accomplished through either the local console itself, that is, physically sitting in front of the machine, or through a terminal session over the network. This network session obviously needs to be encrypted. Zorp uses the industry–standard SSH protocol to accomplish this. SSH is a client–server protocol that provides an encrypted communication channel between the parties. Zorp contains a native SSH server implementation, and SSH clients are freely available for most major operating systems. If you do not have one already installed on your system, visit http://www.freessh.org for a list of both free and payware SSH clients.

To establish an SSH session, the client must first authenticate itself to the server. A number of different authentication methods are defined in the SSH protocol standard. Currently, SSH version 2 is the latest standard version.

The simplest authentication method is password authentication: the user logging in to the SSH server provides a username/password pair that is a valid user in the server machine. The advantage of this method is its simplicity: no special configuration is needed and the user must know the username and password on the server anyway.

A more secure method of SSH login is the public key-based authentication: in this case the user possesses a public/private key pair and the server has a copy of the user's public key. The logon procedure using public keys is the following:

1. Using the private key, the user initiates a logon to the server with the help of a session identifier.
2. The server checks whether it has the matching public key for the user and grants access if both the key is found and the signature is correct.

The advantage of this method is enhanced security. No username and password have to be provided and the entire authentication session is secured with public key cryptographic procedures. The disadvantage is the extra setup needed: you have to generate user key pairs, place public keys of users on the server and the user has to carry the private keys to all client computers the user wishing to log on from, and upon leaving the client, the user has to make sure to remove the private key from the computer. Although, the more complicated it is to set up this method of authentication, the more preferred it is, due to the enhanced security it provides.

By default, Zorp allows password-based SSH logins, too. During the setup it has to be decided whether the root user can log in to the system through SSH connection. It is recommended not to allow roots to login through SSH for security reasons. First an SSH session has to be established as a normal user and then perform a `sudo` action to gain special access permission for accomplishing administrative tasks.

For more information on SSH and its configuration, see *Appendix C, Further readings (p. 485)*.

10.3. Editing configuration files

Local system configuration is performed by editing the appropriate configuration files and then reloading or restarting the corresponding service(s).

All important system components, such as daemons, services, have their own configuration files, some have more than one. These files are generally stored under the `/etc` directory. There are exceptions from this rule, of course, but the majority of configuration files are in that directory. Some files are directly stored under `/etc`, but most services store the configuration files for the services in a subdirectory. For example, Zorp stores a number of configuration files under `/etc/zorp`.

The configuration files are usually plain-text ASCII or XML files, and can be edited with any text editor. By default, the following text editors are installed: `joe`, `nano`, and `vi`.

Tip
Before editing configuration files make backup copies, for example, using the following command:`cp filename filename.bak`

Warning
Zorp replaces the configuration file of several services with a symbolic link that points to a configuration file that is maintained by Zorp. Do not edit such files directly, because the changes will be automatically removed at the next upgrade to a new version of Zorp.

To edit such files properly, first break the symbolic link, and replace the broken link with a file. Following that, the file can be edited. The list of files replaced with symbolic links is the following:`/etc/apparmor.d/abstractions/base_reduced`,

/etc/apparmor.d/abstractions/nameservice, /etc/default/spamassassin, /etc/default/snmpd, /etc/dhcp3/dhclient.conf, /etc/grub.d/10_linux/etc/openvpn/up.py, /etc/init/procps-late.conf, /etc/init.d/kdump, /etc/ip6tables.conf.in, /etc/ip6tables.conf.var, /etc/logrotate.d, /etc/network/if-up.d/group , /etc/network/if-up.d/dhcp3-relay, /etc/openswan/ipsec.conf, /etc/rc.local, /etc/syslog-ng/syslog-ng.conf

10.4. Network configuration

Apart from special setups where you need to fine-tune various performance parameters, the network configuration is a relatively simple task under Zorp. You have to provide basic IP parameters, such as IP addresses, subnet masks, default gateway. The most important configuration file for networking is `/etc/network/interfaces`. This file contains separate sections for all network interfaces available in the given system. The official installation procedure of Zorp involves steps to configure these basic IP parameters, so in a properly installed system this file is not empty.

Note
After the first upload of a configuration file edited in ZMC, the following three-line comment is displayed at the beginning of all editable files under ZMC:

```
#
# This file is generated by Zorp Management Server. Do not edit!
#
```

This warning reminds on that even if a file is edited manually, it is overwritten at the next upload of any change in ZMC. This warning can be ignored safely though in case the edited file is not planned to be used from ZMC in the future.

A typical interface configuration section in `/etc/network/interfaces` is the following.

```
auto eth0
iface eth0 inet static
 address 192.168.1.253
 netmask 255.255.255.0
 broadcast 192.168.1.255
 network 192.168.1.0
 gateway 192.168.1.254
```

After editing and saving this file, activate the changes by running the `/etc/init.d/networking` script with the `restart` argument. This applies to all other network configuration files, too.

Another, less frequently modified file of network configuration is `/etc/hostname`. It contains the hostname parameter of the system. It is important for the name resolution processes initiated by various system components. Whenever a process needs name resolution, that is, to map a name to an IP address, the first thing the system does, is that it checks this file to see whether its own hostname has been queried. The Linux command hostname queries this file as well.

The file / `etc/mailname` is important for the proper operation of the Postfix native service. It must not be empty and it gets filled automatically. You can alter the value stored here, if needed.

Another network configuration file, `/etc/hosts` may be used for static name resolution: it stores name to IP address mappings for network hosts. Before the DNS solution, this file was the only means to map hostnames to IP addresses. Today, most of its functionality has been taken over by DNS, but it is still useful in some scenarios. When a hostname needs to be looked up, `/etc/hosts` is the third place the system looks for a match – the first is `/etc/hostname` while the second is the in-memory DNS cache. Therefore, if there is a limited number of hosts the firewall often visits, among which there is, for example, a proxy server, it is recommended to list these hosts in `/etc/hosts`:

```
#
# This file is generated by Zorp Management Server. Do not edit!
#
127.0.0.1 localhost
192.168.1.253 proxy
192.168.1.100 mail
```

By default, there is only one entry in this file for the hostname localhost with the IP address 127.0.0.1. This entry is needed for system boot processes, therefore it shall not be deleted.

Similarly to hostnames, networks can be named with symbolic names. The file `/etc/networks` stores these mappings. By default, this file is empty on the firewall and Zorp generally does not use it.

The `/etc/resolv.conf` file is used by the resolver library to find what DNS servers to query when a process needs to look up an IP address for a given hostname, or vice versa. In other words, this file lists the known nameservers for the firewall. Additionally, it contains an entry for the domain name of the firewall. This entry is also important for name resolution purposes: if, instead of a fully qualified domain name (FQDN) only a hostname is queried, the resolver automatically appends this domain name to the hostname and tries to look up the FQDN created this way.

```
#
# This file is generated by Zorp Management Server. Do not edit!
#
domain example.org
nameserver 192.168.1.200
```

This section introduces only briefly the network configuration files. For more detailed information and instructions on network configuration, see *Chapter 5, Networking, routing, and name resolution (p. 63)*, the references listed on networking in *Appendix C, Further readings (p. 485)*, and the manual pages for the mentioned files (man filename – without full path, for example: man interfaces).

10.5. System logging

Syslog-ng is the native and recommended logging service for Zorp. Its configuration is stored in the `/etc/syslog-ng/syslog-ng.conf` file.

For more detailed information and instructions on system logging, see chapter *Syslog-ng*, the Syslog reference manual accessible from *Appendix A*, and the installed manual pages for both syslog-ng (the utility) and syslog-ng.conf (the configuration file).

After editing and saving the `syslog-ng.conf` file manually, restart the service by running the `/etc/init.d/syslog-ng` script with the `restart/reload` arguments. Its default configuration under Zorp routes all relevant system, ISC BIND 9 and NTP messages to the `/var/log/messages` file and also to the console, `/dev/tty8`.

If you have a separate, central syslog-ng server for collecting messages from critical network hosts, such as the firewall(s), you can route (log) messages using the following steps.

1. Set up a new destination of TCP or UDP type, with the IP address of your syslog-ng server, in `syslog-ng.conf` on the firewall.

   Example 10.1. Specifying the target IP address of a TCP destination
   The IP address of the syslog-ng server is 10.20.30.40 in this example.

   ```
   destination d_tcp { tcp("10.20.30.40" port(1999); localport(999)); };
   ```

   Supplying port information is optional; if port number is not set, the default ports are used.

2. Decide what sources (s1, s2 here) shall be logged to the syslog-ng server and set up a log path accordingly. Note that filters are optional.

   ```
   log { source(s1); source(s2); filter(f1); destination(d_tcp); };
   ```

3. Define a source on the syslog-ng server with the IP address of the firewall sending log messages.

   Note
   Specify the port numbers carefully. The corresponding ports must match on both sides.

10.6. NTP

The Network Time Protocol (NTP) is used for synchronizing system time with reliable time servers over the Internet. The synchronization is performed by a dedicated service, the NTP daemon (ntpd). The configuration file of ntpd is the `/etc/ntp.conf` file. The `ntp.conf` configuration file is read at initial startup by the NTP daemon in order to specify the synchronization sources, modes and other related information.

Unlike the system logging and network configuration files, `ntp.conf` does not have a manual page in the default installation of Zorp. However, there are many useful sources available on NTP, see for details *Chapter 9, Native services (p. 218)*, the website of NTP protocol/service link in *Appendix C, Further readings (p. 485)*, *RFC 1305* on NTP version 3, and the manual page of `ntp.conf` for accessing it on other Unix/Linux installations or on the Internet.

NTP itself can have a very sophisticated configuration with, for example, public key authentication, access control, or extensive monitoring options. At the very minimum, define a time server with which the firewall can synchronize time (the server key).

Add the following line in the configuration file.

```
server 10.20.30.40
```

Note
If more than one timeserver is supplied, the system time is more accurate, because during a time update all the listed servers are queried and a special algorithm selects the best (most accurate) of them.

Additionally, since Zorp can be used as an authentic time source for the network, you can limit the number of concurrent client connections using the `clientlimit` key, and you can set a minimum time interval a client can synchronize time with the firewall using the `clientperiod` key.

After editing and saving the `ntp.conf` file manually, restart the service by running the `/etc/init.d/ntp` script with the restart argument. NTP can be chrooted as well, in which case the place of the configuration file is `/var/chroot/ntp/etc/`. The configuration can be edited here directly or else the original configuration file can also be used. In the latter case the jailer script updates the configuration inside the chrooted (jailed) environment. The jailer update process involves the following three steps:

1. The original configuration file is modified.
2. Jailer is run.
3. The process (daemon) is restarted.

If ZMC is used for system configuration, the configuration files are automatically created inside the chrooted environment, so no special intervention is needed.

This method for updating jailed environments is the same for all other daemons that are to be jailed under Zorp, such as ISC BIND 9.

System time is updated with the `ntpdate` command. Run the command as root, as usually from a system startup script so that system time gets adjusted during bootup. You can run the command manually, if needed.

10.7. BIND

BIND 9 is the official DNS server solution in Zorp. BIND under Zorp always runs in a chrooted environment, so its configuration file(s) are stored under the `/var/chroot/bind9/etc/bind/` directory.

BIND 9 introduced the notion of split-DNS installations where basic access control can be applied to DNS Zone records. That is, for each record in the DNS database file you can specify whether outside resolvers can query those records ('public' records in DNS terminology) or they are only available to internal resolver clients ('private' records).

Choosing split-DNS setup is optional. In this case there are two configuration files:

- `named.conf`, and
- `named.conf.shared`.

The `named.conf.shared` file hosts information that is intended to be public, that is, accessible to outside resolvers.

Tip
Setting up a split-DNS configuration is reasonable to be used if the firewall is going to be an authoritative nameserver for one or more domains. If it is only used as a forward-only server, split-DNS is not necessary.

In forward-only configurations, only the `named.conf` file is used. Being a forward-only server, the nameserver under Zorp does not perform recursive name resolution on the Internet for the internal clients, but instead, it forwards queries as is to the nameserver(s) configured as its forwarder(s).

This is probably the simplest functional named configuration possible, as only a single entry has to be edited in the configuration file.

```
forwarders {
  10.20.30.40; //IP address of the forwarder nameserver
};
```

You can use BIND 9 as a slave nameserver. In this setup, you do not maintain zone information on the firewall, instead, pull zone database records from an authoritative master nameserver through the zone transfer process. This setup provides fault tolerance, since if the master nameserver fails, the slave still contains a more or less up-to-date copy of the zone database.

The daemon running the BIND service is called named (hence the name for the configuration file), but the directory of the configuration file is still called `bind.conf`, after the name of the original Berkeley Unix implementation of the service. The startup script for the service is also called `/etc/init.d/bind9`.

For further information on BIND, see *Chapter 9, Native services (p. 218)*, and the references listed in *Appendix C, Further readings (p. 485)*.

10.8. Procedure – Updating and upgrading your Zorp hosts

Zorp uses the `apt` package manager application to keep the system up-to-date. Security and maintenance updates as well as product upgrades are all performed with `apt`. To update any host of your Zorp Firewall solution (including Zorp firewall hosts, ZAS and ZCV hosts, as well as your ZMS server), complete the following steps.

For more information on `apt`, see the `apt-get` manual page.

Step 1. Update the `apt` sources of the host. Use one of the following methods:

- **To upgrade from a DVD-ROM**:
  1. Open the *Balasys website*. To open it, it is necessary to authenticate with your support user credentials.
  2. Choose the necessary, preferably the latest version of the ISO file, and download it from the relevant *cd* directory.
  3. Burn the DVD-ROM to physical media.
  4. Mount the DVD-ROM on the host, and execute the following command as root: `>:~#apt-cdrom add`
- **To upgrade from the official Balasys `apt` repositories from the Internet**:
  1. Edit the `/etc/apt/sources.list.d/zorp.list` file and add the URLs of the package sources to download.

For details on the required sources, see *Procedure 4.2, Upgrading Zorp hosts using apt* in *Zorp Professional 7 Installation Guide*.

Warning
Do not remove Ubuntu sources from the `/etc/apt/sources.list` file. These are necessary for upgrading the base operating system.

Step 2. Enter the following two commands.

```
>:~#apt update
```

```
>:~#apt dist-upgrade
```

The first command updates the package list with the latest available versions. The second command performs the upgrade itself.

Step 3. The rest of the process is done automatically.

10.9. Packet filter

The packet filter configuration is stored in the `/etc/iptables.conf` file. Although it is technically possible to edit this file manually, it is not recommended to do so as the first two comment lines of the file warn as well even if manual configuration is chosen over ZMC-based graphical work.

```
#
# This file is generated automatically from iptables.conf.in and iptables.conf.var.
# Do not edit directly, regenerate it using iptables-gen.
```

To make packet filter configuration more error-resistant and easier, a frontend utility pack, the `iptables-utils` has been created where a couple of scripts help the creation and maintenance of packet filter rulesets. For more details on the `iptables-utils`, see chapter *Packet Filtering*.

Tip
Using iptables-utils is absolutely beneficial in the long term as the number of system closeouts -that is administrator lock-outs happen for example by activating an incorrect packet filter ruleset- can be dramatically decreased. It is especially favourable if the administrator is far away from the firewall.

After installing the firewall a default ruleset is active. Since Zorp acts as a default-deny firewall, the ruleset allows only connections from the ZMS host machine specified during installation to the firewall and the outgoing connections originating from the firewall itself. Besides the `iptables.conf` file which stores the currently active ruleset, the `iptables.conf.in` file is also present in the system (`/etc/iptables.conf.in`). For checking the differences between the two files in details, see *Appendix A, Packet Filtering (p. 462)*. The `/etc/iptables.conf.var` file is also stored containing a single statement.

```
#define ZMSHOST <ip_address>
```

This entry allows you to refer to the ZMS host machine by the name ZMSHOST rather than by its IP address when editing the `iptables.conf.in` file. These tools and the intermediate configuration files greatly help the administration of packet filter rulesets. However, an in-depth knowledge of iptables is still needed for the successful management of the packet filter.

For more information, see *Appendix A, Packet Filtering (p. 462)* on Zorp-specific configuration of IPTables, the installed manual pages of iptables (userland utility), and the documentation of Netfilter/IPTables project including a detailed tutorial and HOWTO documents accessible from *Appendix C, Further readings (p. 485)*.

10.10. Zorp configuration

The networking configuration of the firewall which involves IP addresses, hostnames, and resolver configuration, rarely changes. However, the daily administration of the firewall often requires the changing of the actual ruleset. For more information on this process, see section *Creating Zorp Policies*.

Basically, the process can be divided into the following two main parts.

1. Configuring the necessary service definition(s).
2. Creating the matching packet filter ruleset, that is generating a skeleton.

The latter packet filter manipulation procedure is detailed in *Section 10.9, Packet filter (p. 243)*. This section shows how to edit a service definition locally.

The key configuration files needed are stored in the `/etc/zorp` directory. The following files play the most important roles in the configuration.

- `policy.py`
  containing complete service definitions

- `instances.conf`
  listing the instances used in the firewall together with their parameters

Tip
In the default installation of Zorp there are two commented sample files, `policy.py.sample` and `instances.conf.sample` that are helpful in getting started with configuration.

To learn command-line policy management it is advised to first use ZMC to graphically generate test-policies and then to check the generated policy files through a terminal connection.

For background information on the possible contents of these files, see *Chapter 6, Managing network traffic with Zorp (p. 87)*.

The configuration of Zorp is based on the Python programming language. The configuration file (`policy.py`) is a Python module in itself. This does not mean, however, that proficiency is required in Python, knowing the syntax of the language and a few semantic elements is sufficient. Though the configuration file may not seem like a complete Python module, it is important to know that it is parsed as one. The following syntactical requirements of Python apply:

Indentation is important as it marks the beginning of a block, similar to what curly braces ('{}') do in C/C++/C#/Java. This means that the way blocks are intended, must be consistent for that given block. The below example shows a correct syntax first followed by an incorrect syntax.

Correct:

```
if self.request_url == 'http://www.balasys.hu/':
  print ('debug message') return HTTP_REQ_ACCEPT
return HTTP_REQ_REJECT
```

Incorrect:

```
if self.request_url == 'http://www.balasys.hu/':
   print ('debug message')
  return HTTP_REQ_ACCEPT
return HTTP_REQ_REJECT
```

Getting used to correct indentation is probably the most important Python task for a beginner, especially without any C or C-like programming experiences. Indentation in Python is the only way to separate blocks of code since there are no Begin and End statements or curly braces. Otherwise, the language itself is quite simple and easy to learn. Note that Python is case-sensitive.

For more information on Python, see *Appendix C, Further readings (p. 485)*.

10.10.1. Policy.py and instances.conf

The `Policy.py` file has a strict structure that must be obeyed when modifying the configuration manually. It consists of the following code modules:

- Import statements
- Zone definitions
- Class configurations
- NAT policy settings
- Authentication policy settings
- Instance definitions

These modules are of varying length, depending on the complexity of the policy configuration.

10.10.1.1. Procedure – Edit the Policy.py file

Step 1. Set the import statements.
The default-installed `policy.py.sample` file starts with the import statements:

```
from Zorp.Core import *
from Zorp.Plug import *
from Zorp.Http import *
from Zorp.Ftp import *
```

These statements mean that one or more required (Python) front-end modules are imported to the configuration. Zorp.Core is essential, however, the other three imports are included because the sample file contains references to these three proxy classes.

Tip
A good way of learning `policy.py` is to create firewall policies in ZMC and then look at the resulting configuration files.

Step 2. Provide the name of the firewall, and the zone definitions along with the access control defined for them, that is, the allowed outbound and inbound services.

```
Zone("site-net", ["192.168.1.0/24"])
```

Step 3. Configure the classes used in service definitions.
These class definitions can be simple, with, in essence, naming the proxy class to be used, that is, to be derived from only; like the IntraFtp class in the sample file:

```
class IntraFtp(FtpProxy):
  def config(self):
    FtpProxy.config(self)
```

Or, they can be rather complex, customizing the derived proxy class with attributes, as in the case of the IntraHttp class in the sample file:

```

# Let's define a transparent http proxy, which rewrites the
# user_agent header to something different.
#
class IntraHttp(HttpProxy):
  def config(self):
    HttpProxy.config(self)
    self.transparent_mode = TRUE
    self.request_headers["User-Agent"] = (HTTP_HDR_CHANGE_VALUE,
"Lynx/2.8.3rel.1")
    self.request["GET"] = (HTTP_REQ_POLICY, self.filterURL)
    # self.parent_proxy = "proxy.site.net"
    # self.parent_proxy_port = 3128
    # self.timeout = 60000
    # self.max_keepalive_requests = 10

  def filterURL (self, method, url, version):
    # return HTTP_REQ_REJECT here to reject this request
    # change self.request_url to redirect to another url
    # change connection_mode to HTTP_CONNECTION_CLOSE to
    # force kept-alive connections to close
    log("http.info", 3, "%s: GET: %s" % (self.session.session_id, url))
```

Step 4. Define the instances to be used.
Besides its name, the most important characteristic of an instance is the list of services it provides. Therefore, define services within the instances:

```
# zorp_http instance
def zorp_http () :
  # create services
  Service(name='intra_http', router=TransparentRouter(),
chainer=ConnectChainer(), proxy_class=IntraHttp, max_instances=0,
max_sessions=0, keepalive=Z_KEEPALIVE_NONE)
  Service(name='intra_ftp', router=TransparentRouter(),
chainer=ConnectChainer(), proxy_class=IntraFtp, max_instances=0,
max_sessions=0, keepalive=Z_KEEPALIVE_NONE)
  Rule(proto=6,
    dst_port=80,
    service='IntraHttp'
    )
  Rule(proto=6,
    dst_port=21,
    service='IntraFtp'
    )
```

Still within the instance definition code block, with correct indentation, specify the firewall rules that will start these services.

These blocks, the zone definition, proxy class definition, instance definition, service definitions, and rule definitions make up the `policy.py` file. The provided example is simple, yet it provides a lot of information on the correct syntax and on the possible contents of the `policy.py` file.

The other configuration file, `instances.conf` is much more simple: it lists the instances to be run, and supplies some runtime arguments for them such as log level. The only compulsory argument for running an instance is the name of the Python file containing the corresponding instance definition. Although the example uses a single policy file (`policy.py`) to store all definitions, it is possible to separate the policy to different `.py` files if it makes maintenance or archiving easier.

In the following example instance definitions are separated into two files, `policy-http.py` and `policy-plug.py`:

```
#instance arguments
#zorp_http --verbose=5 --policy /etc/zorp/policy-http.py
#zorp_plug --policy /etc/zorp/policy-plug.py
```

For more information on the configuration files, see the manual pages for `instances.conf` and Zorp. The manual pages can be accessed by using the `man instances.conf` and `man zorp` commands, installed by default on Zorp. Also see the *Appendix C, Zorp manual pages* in *Zorp Professional 7 Reference Guide* for further details.

10.10.2. Zorp control

Starting and stopping firewall instances is performed automatically using the default installation. However, it is possible to control manually the firewall instances with the `zorpctl` utility. `Zorpctl` starts and stops Zorp instances using the `instances.conf` file. One or more instance names can be passed to `zorpctl` as arguments. If an error occurs while starting or stopping one of them, an exclamation mark ('?') is appended to the name of the instance as the script processes requests.

To control Zorp with `zorpctl`, enter the following lines.

```
zorpctl start|stop <instance name> <instance name> <...>
```

Besides the start and stop parameters for controlling instances, `zopctl` has some other parameters as well.

- *`zorpctl status <instance>`*
  printing the status of the specified Zorp instance

- *`zorpctl szig <instance>`*
  displaying some internal information about the specified Zorp instance

- *`zorpctl inclog <instance>`*
  incrementing the logging level of the specified instance with one level

- *`zorpctl declog <instance>`*
  decrementing the logging level of the specified instance with one level

For a full list of the available parameters with short explanations type `zorpctl` at a command prompt, without parameters, or issue the `man zorpctl` command. The manual page of zorpctl is also available at zorpctl(8) in *Zorp Professional 7 Reference Guide*.

10.11. Managing core dump files

Zorp uses *systemd-coredump* to capture application firewall crashes. *systemd-coredump* provides the `coredumpctl` command line tool to manage core dumps. It creates compressed core dump files under `/var/lib/systemd/coredump` directory, the related logs into the journal, and the syslog-ng `/var/log/messages` file.

The `coredumpctl list` presents a summary about the core dump events. The `coredump dump <PID> -o /path/to/uncompressed/coredump` command line extracts a core dump to a given file referenced by the process's PID.

By default, *systemd-coredump* uses maximum 10% of the underlying storage of */var/lib/systemd/coredump* directory, but leaves at least the 15% free space. The preferred setting ensures more than 200 GByte storage under the filesystem holding the */var/lib/systemd/coredump* directory for the host. For smaller partitions it is recommended to customize *systemd-coredump* settings with the provided *FreeText* template and to define more space than 10% for the core dump usage and more than 15% free space with the help of the *`MaxUse`*`=` and *`KeepFree`*`=` parameters.

Chapter 11. Key and certificate management in Zorp

The use of cryptography, encryption and digital signatures is becoming more and more widespread in electronic communication. Nowadays they are an essential part of e-business and e-banking solutions, as well as other fields where the identity of the communicating parties has to be verified. Communication through secure (encrypted) channels is also becoming increasingly popular. This chapter offers a brief introduction into the fields of cryptography and the public key infrastructure (PKI), describing how they can be used for authentication and secure communication, and the PKI system developed for ZMS to support them.

11.1. Cryptography basics

The goal of using encryption in communication is twofold: to guarantee the privacy of the communication, so that no third party can acquire the information, and to verify its integrity — to make sure that it was not damaged (or deliberately modified) on the way. Privacy can be guaranteed by the use of encryption algorithms, while integrity protection requires the application of hashing algorithms. Secure communication utilizes both of these techniques.

The main concepts and requirements of secure communication are the following:

- Both the sender and the receiver of the message have access to the algorithm (this is practically a piece of software, often used transparently to the actual user) that can be used to encrypt and decrypt the message.
- Both have access to a special piece of information — so called key — that is required to encrypt and to successfully decrypt the message.
- An encrypted message cannot be decrypted without the proper key, even if the encryption algorithm is known.
- The receiver can identify if the encrypted message has been damaged or modified. (Remember, it is not necessary to understand the message to mess it up.)
- Both the sender and the receiver can verify the identity of the other party.

11.1.1. Symmetric and asymmetric encryption

There are two main categories of encryption methods for ensuring privacy: symmetric and asymmetric encryption.

11.1.1.1. Symmetric encryption

Symmetric encryption algorithms use the same key for the encryption and decryption of a message, therefore the same key has to be available to both parties. Their advantage is their speed, the problem is that the key has to be transferred to the receiver somehow. The keys used in symmetric encryption algorithms nowadays are usually 128-256 bit long.

11.1.1.2. Asymmetric encryption

Asymmetric encryption methods use different keys for the encryption and the decryption of a message. The sender generates a keypair, messages encrypted with one of these keys can only be decoded with the other one. One of these keys will be designated as the private key, this will be used to encrypt the messages. The other key, called public key is made available to anyone the sender wishes to send messages to. Anyone having access to the encrypted message and the public key can read the encrypted message and be sure that it was created with the appropriate private key. Certain encryption algorithms (like RSA) make it also possible to encrypt a message using the public key, in this case only the owner of the private key can read the message. The disadvantage of asymmetric encryption is that it is relatively slow and computation intensive. A suitable infrastructure for exchanging public keys is also required; this is needed to verify the identity of the sender, confirming that the message is not a forgery. This topic is discussed in *Section 11.1.1.3, Authentication and public key algorithms (p. 250)* The length of the keys used in asymmetric encryption ranges from 512 to 4096 bits.

Tip
It is recommended to use at least 1024 bit long keys.

11.1.1.3. Authentication and public key algorithms

Being able to decrypt a message using the appropriate public key guarantees only that it was encrypted with its matching private key. It does not mean that the person (or organization) who wrote the message is who he claims to be — that is, the identity of the sender cannot be verified this way. Without an external way to successfully verify the identity of the other party, this would make communication based on public key algorithms susceptible to man-in-the-middle attacks. To overcome this problem, the identity of the other party has to be confirmed by an external, trusted third party. Two models have evolved for that kind of identity verification: web of trust and centralized PKI.

Web of trust and centralized PKI

In a web of trust based system (such as PGP), individual users can sign the certificate (including the public key and information on the owner of the key) of other users who they know and trust. If the certificate of a previously unknown user was signed by someone who is known and trusted, the identity of this new user can be considered valid. Continuing this scheme to many levels, large webs can be built. Web of trust does not have a central organization issuing and verifying certificates — this is both the strength and weakness of such systems.

In centralized PKI — as its name suggests — there are certain central organizations called Certificate Authorities (CAs) empowered to issue certificates. Centralized PKI systems are described in detail in *Section 11.2, PKI Basics (p. 252)*.

11.1.1.4. Usage of encryption algorithms for secure communication

In real-world communication, the two types of encryption are used together: a (symmetric) session key is generated to encrypt the communication, and this key is exchanged between the parties using asymmetric encryption.

The general procedure of encrypted communication is the following:

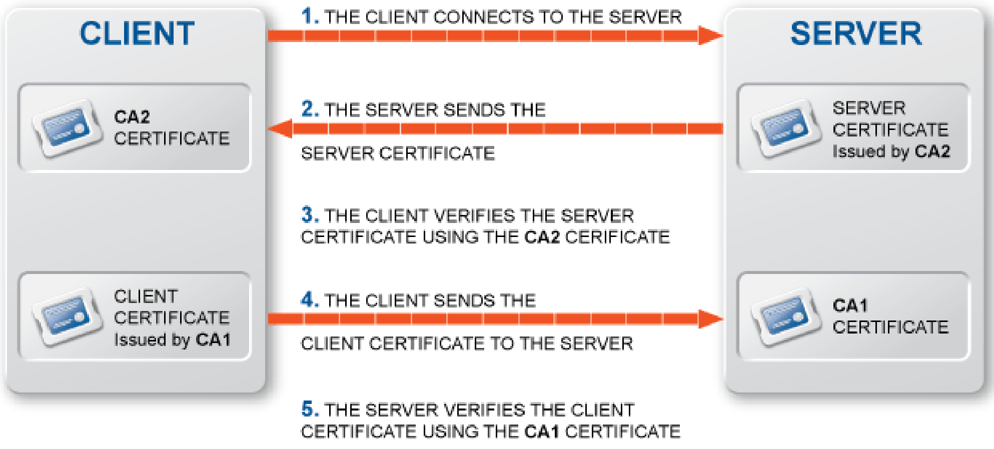

CLIENT
CA2 CERTIFICATE
CLIENT CERTIFICATE Issued by CA1
SERVER
SERVER CERTIFICATE Issued by CA2
CA1 CERTIFICATE
1. THE CLIENT CONNECTS TO THE SERVER
2. THE SERVER SENDS THE SERVER CERTIFICATE
3. THE CLIENT VERIFIES THE SERVER CERTIFICATE USING THE CA2 CERIFICATE
4. THE CLIENT SENDS THE CLIENT CERTIFICATE TO THE SERVER
5. THE SERVER VERIFIES THE CLIENT CERTIFICATE USING THE CA1 CERTIFICATE

Figure 11.1. Certificate-based authentication

11.1.1.4.1. Procedure – Procedure of encrypted communication and authentication

Step 1. The sender and the receiver select a method (encryption algorithm) for encrypting the communication.

Step 2. The sender authenticates the receiver by requesting its certificate and public key. Optionally, the receiver can also request a certificate from the sender, thus mutual authentication is also possible. During the handshake and authentication the parties agree on a symentric key that will be used for encrypting the data communication.

Step 3. The sender encrypts his message using the symmetric key.

Step 4. The sender transmits the message to the receiver.

Step 5. The receiver decrypts the message using a symmetric key.

Step 6. The communication between the parties can continue by repeating steps 3-5.

Another important aspect is that suitable keys have to be created and exchanged between the parties, which also requires some sort of secure communication. It also has to be noted that — depending on the exact communication method — the identity of the sender and the receiver might have to be verified as well.

The strength of the encryption is mainly influenced by two factors: the actual algorithm used, and the length of the key. From the aspect of keylength, the longer the key is, the more secure encryption it offers.

11.1.1.5. Hashing

Hashing is used to protect the integrity of the message. It is essentially a one-way algorithm that is capable of creating a fixed-length extract of the message (or document). This extract (hash) is:

- specific to the given document,
- changing even a single bit in the document changes the hash,
- it is not possible to predict how a certain change in the document modifies the hash (that is, it is impossible to predict the hash),
- it is not possible to recover the original document from the hash.

11.1.1.6. Digital signature

A digital signature is essentially the hash of the signed document, encrypted by the private key of the signer. The genuineness of the document can be verified by generating the hash of the document received, decrypting the signature using the public key of the sender, and comparing the hash contained in the signature to the one generated from the received document. If the two hashes are identical, the document received is the same as the one sent by the sender, and has not been modified on the way.

11.2. PKI Basics

The purpose of PKI system is to provide a way for users to reliably authenticate each other. This requires the users to have private-public keypairs (as described in *Section 11.1.1.2, Asymmetric encryption (p. 250)*), some sort of certificate to verify the users identity, and a system to manage and distribute keys and certificates. For verifying the identity of a user, either centralized PKI systems, or webs of trust can be used.

11.2.1. Centralized PKI system

The centralized model is based on authorizing institutes, so called Certificate Authorities (CA) to verify the identity of the user or organization and certify it in a digital certificate. Since there is no single, worldwide CA guaranteeing the identity of everyone, the identity of a party can be considered valid if its certificate was signed by a trusted CA. A trusted CA is a CA that has been decided to be trustworthy, there is no general algorithm or method to determine which CAs can be trusted. A 'trusted CA list' includes the certificates of all the CAs deemed trustworthy.

11.2.1.1. CA chains and Root CAs

CAs themselves also have to certify their identity, meaning they also need certificates. These certificates are usually signed by another, higher level CA. This allows for hierarchies of CAs to be created, in a way that although a CA might not be explicitly trusted (because it is unknown, therefore it not on the list of trusted CAs), but the higher-level CA that signed its certificate might appear on the list (which makes the lower-level CA trustworthy).

Obviously, using this method alone is not sufficient, since it always requires a higher-level CA. Therefore, self-signed CA certificates also exist, meaning that the CA itself has signed its own certificate. This is not uncommon; a CA with a self-signed certificate is called a root CA, because there is no higher-level CA above it. To trust a certificate signed by this CA, it must necessarily be in the 'trusted CAs' list.

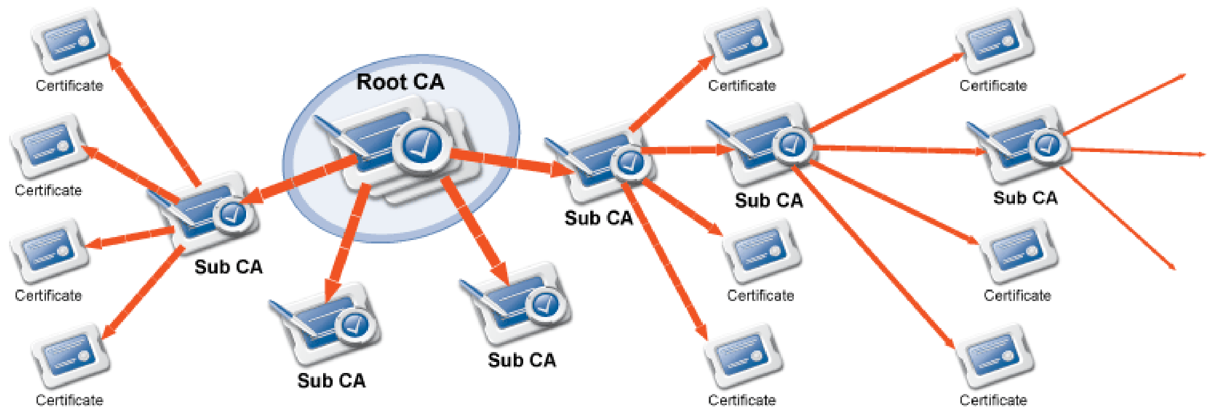


Figure 11.2. Certificate chains

Note
To allow easier management, the trusted CA lists usually contain only root CAs.

11.2.2. Digital certificates

A digital certificate is a digital document conforming to the X.509 standard that certifies that a certain public key is owned by a particular user or organization. This document is signed by a third party (the CA). This data file contains the public key of its owner, as well as the following information:

- **Not before/Not after**: Validity (from/to date) of the certificate.
- **Purpose**: For what end may the certificate be used (for example, digital signature, data encryption, and so on).
- **Issuer**: The Distinguished Name of the Certificate Authority that signed the certificate.
- **Subject**: The Distinguished Name of the owner of the certificate.
- **Distinguished Name**: The distinguished name (DN) usually contains the following information (not all the fields are mandatory, and other optional fields are also possible). A DN is often represented as a comma-separated list of fieldname-value pairs.
  - Country: 2-character country/region code.
  - State: State where the organization resides.
  - Locality: City where the organization resides.
  - Organization: Legal name of the organization.
  - Organizational Unit: Division of the organization.
  - Common Name: The common name is often the address of the website or the domain name of the organization, for example, www.example.com, or the name of the user in case of personal certificates.

11.2.3. Creating and managing certificates

When an organization wishes to create a certificate, it has to perform the following:

11.2.3.1. Procedure – Creating a certificate

Step 1. Generate a private-public keypair.

> **Tip**
> The secure storage of private keys has to be solved.

Step 2. Prepare a certificate signing request (CSR). For filling the request form, the information contained in the distinguished name has to be provided (for example, common name, organization, and so on).

Step 3. The CSR is bundled together with the public key of the generated keypair.

Step 4. The organization selects a CA to sign the certificate request. The CSR has to be submitted to a special department of the CA, called Registration Authority (RA).

Step 5. The RA verifies the identity of the requestor.

> **Note**
> Submission of the CSR to the RA and the identity verification involves physically visiting the RA with all the papers it requires for verifying the identity of the organization and its representative (for example, documents of incorporation, ID cards, and so on).

Step 6. If the RA confirms the identity of the requestor, the CA signs the request using its private key, and issues the certificate.

> **Tip**
> If the certificate is to be used only internally (as in the case of Zorp components), an own CA with a self-signed certificate can be set up to sign the certificates.

Step 7. The requestor can now import and use the certificate on his machines.

Step 8. If a certificate loses its validity or becomes obsolete, it should not be accepted anymore and is to be revoked or refreshed.

Basically the CA has the following functions:

- Checks the identity of everyone requesting a certificate.
- Confirms the identity of a user by its signature.
- Monitors the validity of issued certificates (see *Section 11.2.5.1, Certificate Revocation List - CRLs (p. 255)* below).

> **Tip**
> Although to efficiently use certificates over the Internet they need to be signed by well-known Certificate Authorities, this is not required if they are used only locally within an organization. For such cases, the organization itself can create a local (internal) CA and sign the

certificate of this CA. This CA having a self-signed certificate (thus it becomes the local root CA) can then be used to sign the certificates used only internally.

11.2.4. Verifying the validity of certificates

To decide whether a given certificate is valid or not, the followings have to be checked:

- It was signed by a trusted CA.

  Note
  If the certificate of the CA signing the given certificate was signed by a trusted CA (or by another CA lower in the CA chain), the certificate can be trusted. Sometimes this CA chain can consist of several levels.

- It is not out-of-date.
- It has not been revoked.
- The purpose of the certificate is appropriate, that is, it is used for the issued intention.

Note
It is possible to submit certificate signing requests (CSRs) to more than one CA (and have them signed) using the same public key. However, it is considered to be highly unethical, likely resulting in the revocation of all of the certificates involved.

11.2.5. Verification of certificate revocation state

Zorp supports the following two solutions from the available methods for the verification of certificate revocation state:

- Certificate Revocation Lists (CRLs)
- Online Certificate Status Protocol (OCSP) stapling

Both methods are available for client- and server-side verifications as well in encryption policies.

When setting up and performing revocation checking, the encryption policies do not separate the two methods. If revocation checking is enabled, then Zorp attempts to gain valid revocation information using both methods and uses any valid result.

11.2.5.1. Certificate Revocation List - CRLs

Certificate Revocation List (CRL) is a list containing the serial numbers and distinguished names of certificates that cannot be trusted anymore and were hence revoked. If a certificate loses its validity for any reason (for example, becomes compromised because its private key is stolen) the issuing Certificate Authority (CA) revokes it. This is published on the website of the CA in a CRL. Expired or compromised certificates shall not be used either internally.

CRLs can be obtained usually through HTTP, certificate authorities update and publish them on their website on a regular basis.

11.2.5.2. Online Certificate Status Protocol (OCSP) stapling

Online Certificate Status Protocol (OCSP) stapling is an alternative to the so far available Certificate Revocation Lists (CRL) in verifying the validity of certificates. The protocol is described in details in IETF RFC 6960. With OCSP stapling it is possible to define to what level of strictness, the encryption policies shall check the revocation status of the certificates.

Online Certificate Status Protocol stapling provides the following benefits:

- The solution enables a more convenient solution of assigning server operators to keep revocation information up-to-date instead of requiring that from clients.
- Due to the smaller size of the used traffic data during OCSP stapling compared to CRL processes, the network load is smaller as well.
- Clients can verify the revocation state of a certificate with minor overhead.

OCSP stapling provides a potentially faster revocation state with less traffic. The responsibility of obtaining a certificate revocation state is moved from the client (e.g. web-browser) to the server. The servers fetch revocation information of their certificates and cache this information for a short period of time. When a client attempts to establish a secure connection with the server, the server staples the revocation state to the certificate it is sending to the client.

For more details, see *Section 3.2.4, Configuring Encryption policies* in *Zorp Professional 7 Reference Guide*.

11.2.6. Authentication with certificates

Authentication with certificates is accomplished by checking the validity of the certificates of the communicating parties.

One-way authentication: One of the parties (typically the client) requests a certificate of the server and checks its validity.

Mutual (two-way) authentication: Both the client and the server check the validity of the other's certificate. Generally both parties must own a trusted certificate (that is, a certificate signed by a trusted certificate authority).

11.2.7. Digital encryption in work

SSL provides endpoint authentication and communications privacy, as well as possibility for one-way or mutual authentication using certificates. The protocol allows client/server applications to communicate without being subject to eavesdropping, tampering, or message forgery. SSL runs on layers beneath application protocols (for example, HTTP, SMTP, and so on) and above the TCP transport protocol. SSL is able to use a number of symmetric and asymmetric encryption algorithms. The certificates used in the communication must conform to the X.509 standard.

IPSec is a set of protocols for securing packet flows and key exchange by encrypting and/or authenticating all IP packets. As IPSec is an obligatory part of IPv6 (and optional in IPv4), it can be expected that it will become increasingly widespread. IPSec provides end-to-end security for packet traffic — even for UDP packets, because it operates over the IP layer. In Zorp, IPSec is used to construct Virtual Private Networks (VPNs). Please refer to *Chapter 16, Virtual Private Networks (p. 426)* for more details.

Note
Zorp supports the use of the Secure Sockets Layer (SSLv2 and SSLv3), Transport Layer Security (TLSv1) and IP Security (IPSec) digital encryption protocols.

11.2.8. Storing certificates and keys

When importing/exporting keys and certificates, they can be stored in various file formats. ZMS supports the use of the PEM, DER, and PKCS12 file formats. The main differences between them are summarized below.

- *PEM*: PEM (Privacy Enhanced Mail) is an ASCII text format that can store all parts of the certificate, that is, certificate, certificate signing request (CSR), Certificate Revocation List (CRL), private key (which can be optionally protected with a password). It is not necessary to store all parts in a single file.

  Tip
  If nothing restricts it, it is recommended to use the PEM format.

- *DER*: The DER (Distinguished Encoding Rules) format stores any single part of a certificate in a binary file.
- *PKCS12*: The PKCS12 (Public Key Cryptography Standards) is a binary file format developed to provide an easy and convenient way to backup or transport certificates. The file always contains a password-encrypted private key and the associated certificate.

11.3. PKI in ZMS

The purpose of including a light-weight PKI system in ZMS is to provide a convenient and efficient way to manage and distribute certificates and keys used by the various components and proxies of the managed Zorp hosts. It is mainly aimed at providing certificates required for the secure communication between the different parts of the firewall system, for example, Zorp hosts and ZMS engine (the actual communication is realized by agents). The PKI of ZMS also provides a consistent and convenient tool to manage both internal and external certificates between the firewalls. ZMS can be set to perform the regular distribution of certificates and Certificate Revocation Lists (CRLs) automatically, ensuring that no invalid or revoked certificate can be used.

Note
It has to be noted that the PKI of Zorp is not a general purpose PKI system, consequently it is not recommended to be used as such. It was designed and intended for internal use between the components of the firewall system (to secure the communication between Zorp hosts and ZMS servers, monitoring agents, and so on), and to manage external certificates available on the managed hosts.

Tip
The PKI system of ZMS can also manage certificates signed by external CAs. This is useful because ZMS provides an efficient way to handle the distribution of certificates among the managed hosts.

11.3.1. Committing changes and locking in PKI

When an administrator starts to modify the PKI settings (either on one of the **Edit Certificates** panels or the **Site Preferences**), the PKI component is locked from other administrators. Changes are committed automatically.

11.3.2. The certificate entity

ZMS manages the certificates, their accompanying keys, as well as the related certificate signing request (CSR) and Certificate Revocation Lists (CRL(s)) as a single entity. Therefore when using a key, certificate, CSR or CRL in connection with ZMS, this single entity containing all of them is referred. This is important to remember even if not explicitly stated in the text.

In ZMS, a certificate entity has two different names, these are:

- *Unique name*: The unique name is the name used to unambiguously identify the certificate entity (and its different parts) in ZMS. This name does not appear in the certificate, it is required only for management purposes.
- *Distinguished name*: It is the distinguished name (DN) of the owner of the certificate. (Sometimes only the **Common Name** part is shown.) For more information, see *Section 11.2.2, Digital certificates (p. 253)*.

11.3.3. Rules of distribution and owner hosts

The owner host is the machine allowed to use the private key. (For example, when specifying on a host which certificate should be used for authentication to management agents, only the certificates owned by the given host can be selected.) It is important to set the owner host of a certificate otherwise it would be impossible to use that certificate entity for all purposes (like authentication).

Distribution of certificates can be handled automatically by ZMS. ZMS examines which certificates are used by the given host, and deploys only those. This ensures that certificates are not unnecessarily present on all machines.

Any part of the certificate entity has to be deployed to the proper host in order to be used. Two main rule governs the distribution (deployment) of certificate entities:

- Every certificate entity is distributed only to those hosts that actually use it, and only the used parts are deployed.
- The private key can be used only on the host(s) that are set as the owner host of the certificate entity. (Therefor the private key is only distibuted to the owner host of the certificate entity.)

Note
CAs do not belong to a single host, but to the whole site, therefore their certificate entity (including their private key) can be made available on each host.

Certificates (not the full entity, only the certificate part) can be distributed everywhere.

Warning
Distribution should only be performed for complete, consistent settings. Distributing incomplete or only partially refreshed configuration can lead to lockouts. This is especially true when regenerating the keys of transfer agents. To prevent such situations, it might be useful

to disable the automatic distribution when making large modifications to the PKI system, and re-enable it only after the new configuration is finished.

11.3.4. Trusted groups

Trusted CAs can be organized into so called trusted groups for more convenient use, especially for configuring proxies using certificates for authentication. In ZMS policies, CAs are referenced through the trusted groups containing them.

Tip
The use of trusted groups is useful for example when configuring SSL proxying, especially if connection only to servers having a certificate issued by a well-known and trusted CA (that is, not self-signed) is permitted. For more information on SSL, see *Chapter 3, The Zorp SSL framework* in *Zorp Professional 7 Reference Guide*.

11.3.5. The PKI menu

The PKI system of ZMS can be accessed by using the **PKI** menu of the main menu bar.

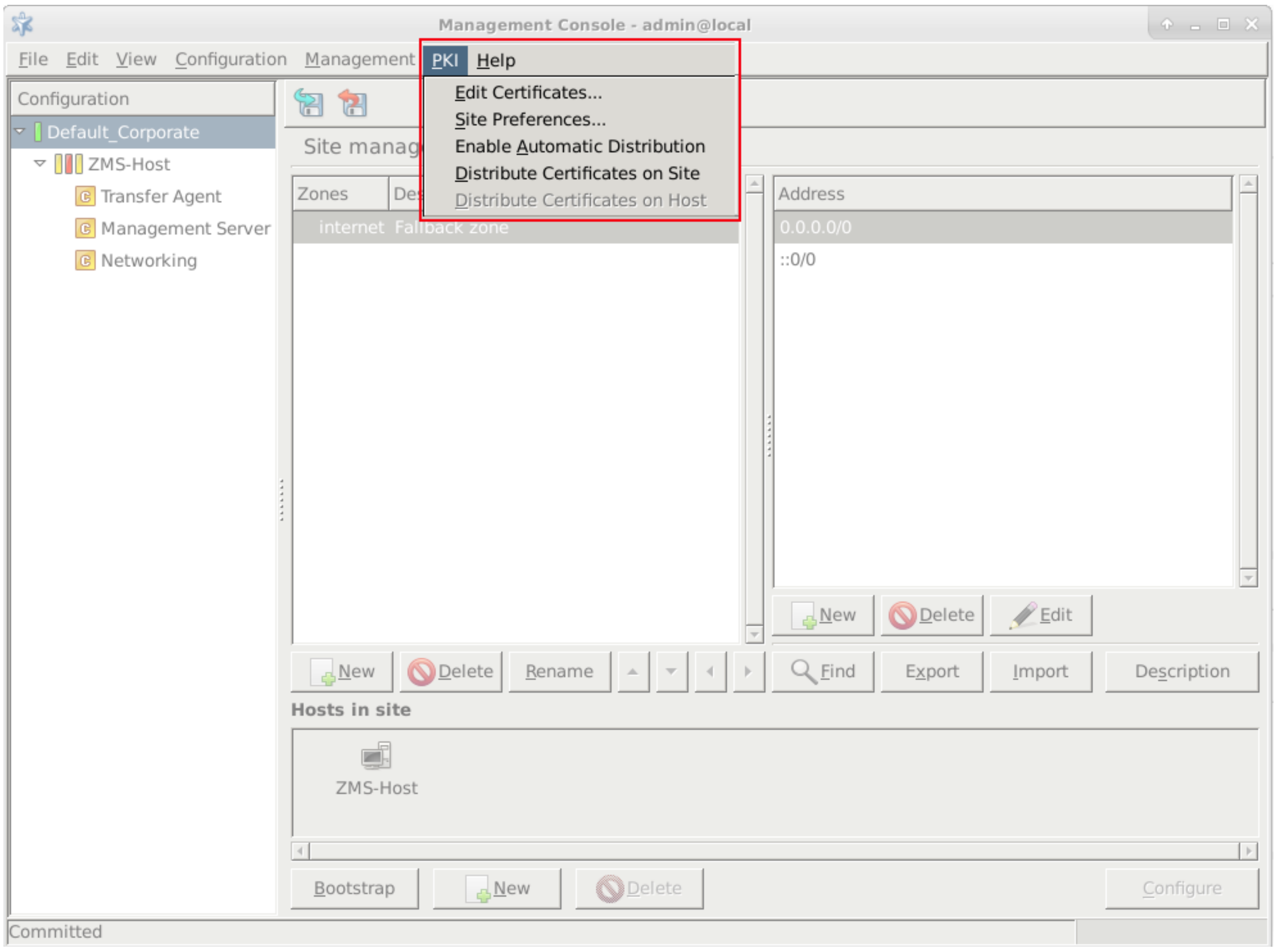
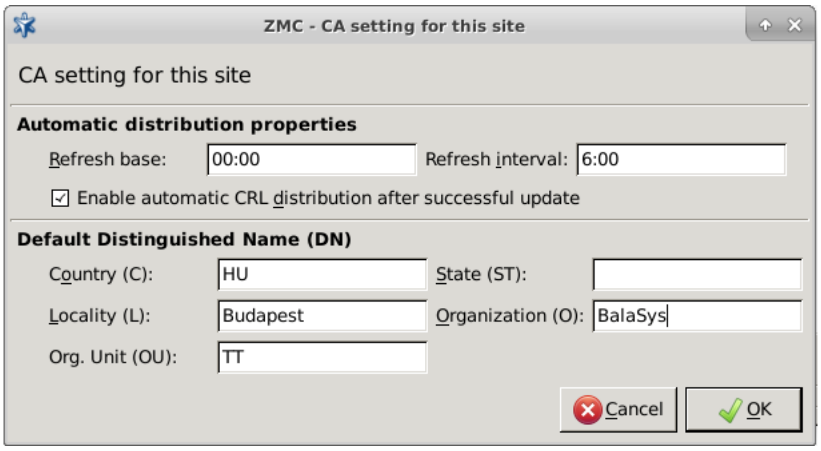


Figure 11.3. The PKI menu

The following sections introduce the function and use of each menu item.

11.3.5.1. Site Preferences

The **Site Preferences** menu can be used to apply site-wide parameters to the CAs.



ZMC - CA setting for this site

CA setting for this site

Automatic distribution properties

Refresh base:	00:00	Refresh interval:	6:00

☑ Enable automatic CRL distribution after successful update

Default Distinguished Name (DN)

Country (C):	HU	State (ST):	
Locality (L):	Budapest	Organization (O):	BalaSys
Org. Unit (OU):	TT		

Figure 11.4. Site Preferences

The parameters in details are:

- **Automatic distribution properties**
  - **Refresh base and refresh interval**: These parameters define the starting time and the interval of the automatic certificate distribution, that is, at what time the distribution of certificates and CRLs should start, and how often it should be performed.

Tip
It is recommended to perform automatic distribution every 4 or 6 hours.

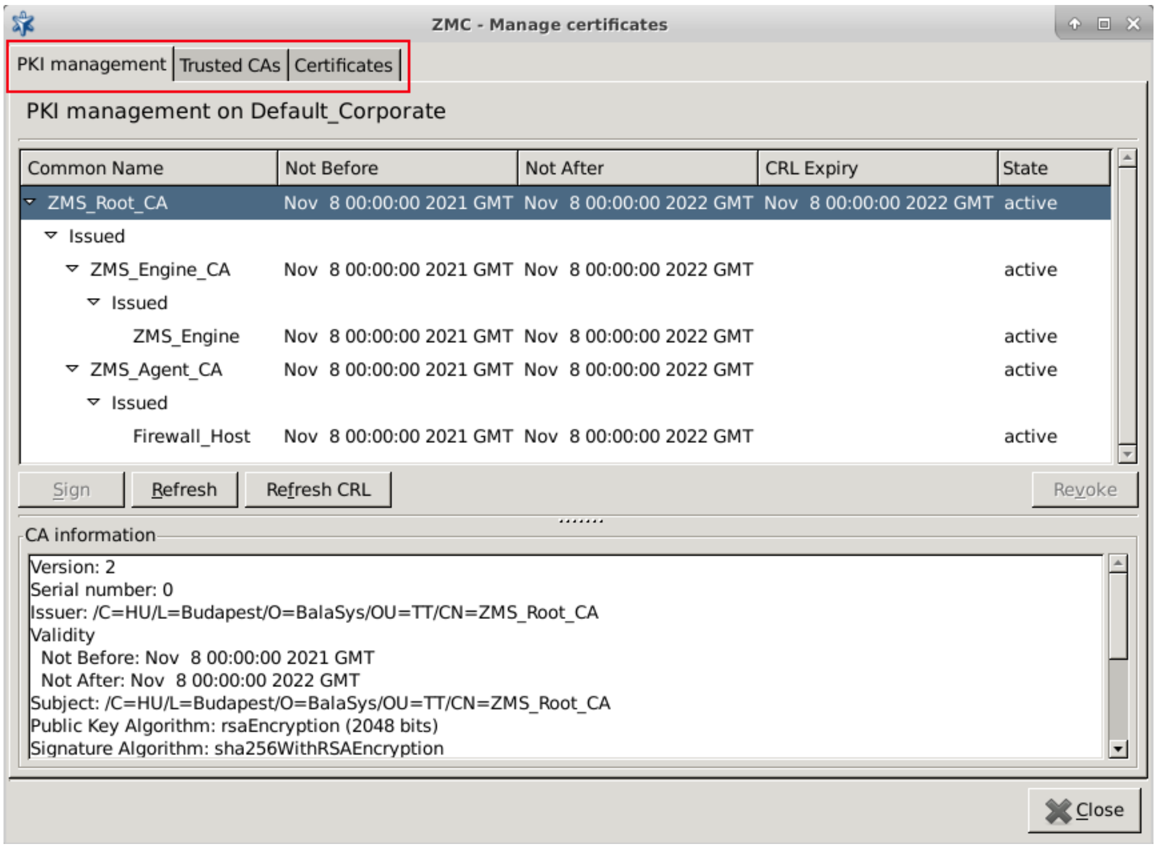
- **Default distinguished name**: The fields of of these parameters will be automatically filled when creating new CA certificates or CSRs, which is especially useful if a large number of certificates has to be created.

11.3.5.2. Distribution of certificates

The automatic certificate distribution can be enabled from the **PKI** menu, and will be performed based on the parameters set under the **Site Preferences** menu item. Manual distribution can be performed by selecting **Distribute Certificates** from **PKI** in the main menu. When distributing CA certificates, the CRLs are also distributed.

11.3.5.3. The Edit Certificates menu

Most of the actual PKI-related tasks can be performed using the **Edit Certificates** menu item. Selecting this item displays the PKI management window of the selected site.



PKI management on Default_Corporate

Common Name	Not Before	Not After	CRL Expiry	State
ZMS_Root_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	Nov 8 00:00:00 2022 GMT	active
Issued				
ZMS_Engine_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
Issued				
ZMS_Engine	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
Issued				
Firewall_Host	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active

CA information

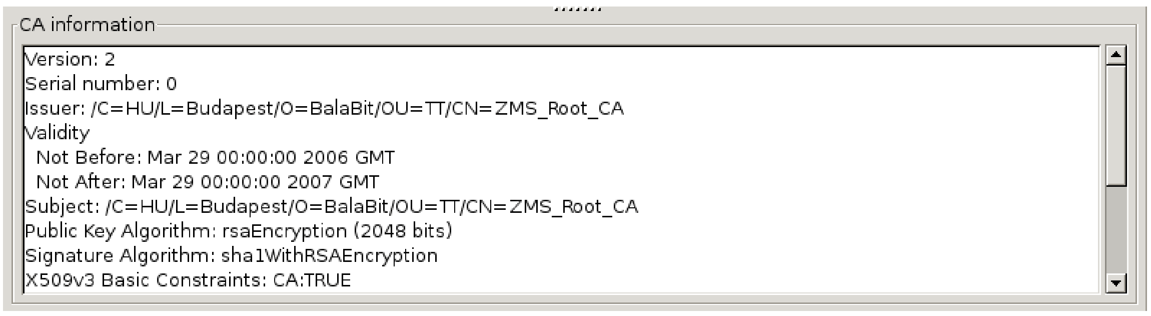
```
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Root_CA
Validity
 Not Before: Nov  8 00:00:00 2021 GMT
 Not After: Nov  8 00:00:00 2022 GMT
Subject: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Root_CA
Public Key Algorithm: rsaEncryption (2048 bits)
Signature Algorithm: sha256WithRSAEncryption
```

Figure 11.5. The Edit Certificates menu

This window has the following tabs:

- ***PKI management*** tab is used for managing local CAs. This includes managing certificates and certificate signing requests, refreshing keys, and so on.
- ***Trusted CAs*** tab is for managing trusted certificate authorities, creating new ones, grouping them, and so on.
- ***Certificates*** tab is for managing certificates: creating new certificate signing requests (CSRs), as well as for importing/exporting certificate entities.

On all three tabs, information about the currently selected certificate (or CA certificate) is displayed in the lower section of the panel. This information includes the following data:


CA information

```
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaBit/OU=TT/CN=ZMS_Root_CA
Validity
  Not Before: Mar 29 00:00:00 2006 GMT
  Not After: Mar 29 00:00:00 2007 GMT
Subject: /C=HU/L=Budapest/O=BalaBit/OU=TT/CN=ZMS_Root_CA
Public Key Algorithm: rsaEncryption (2048 bits)
Signature Algorithm: sha1WithRSAEncryption
X509v3 Basic Constraints: CA:TRUE
```

Figure 11.6. Certificate information

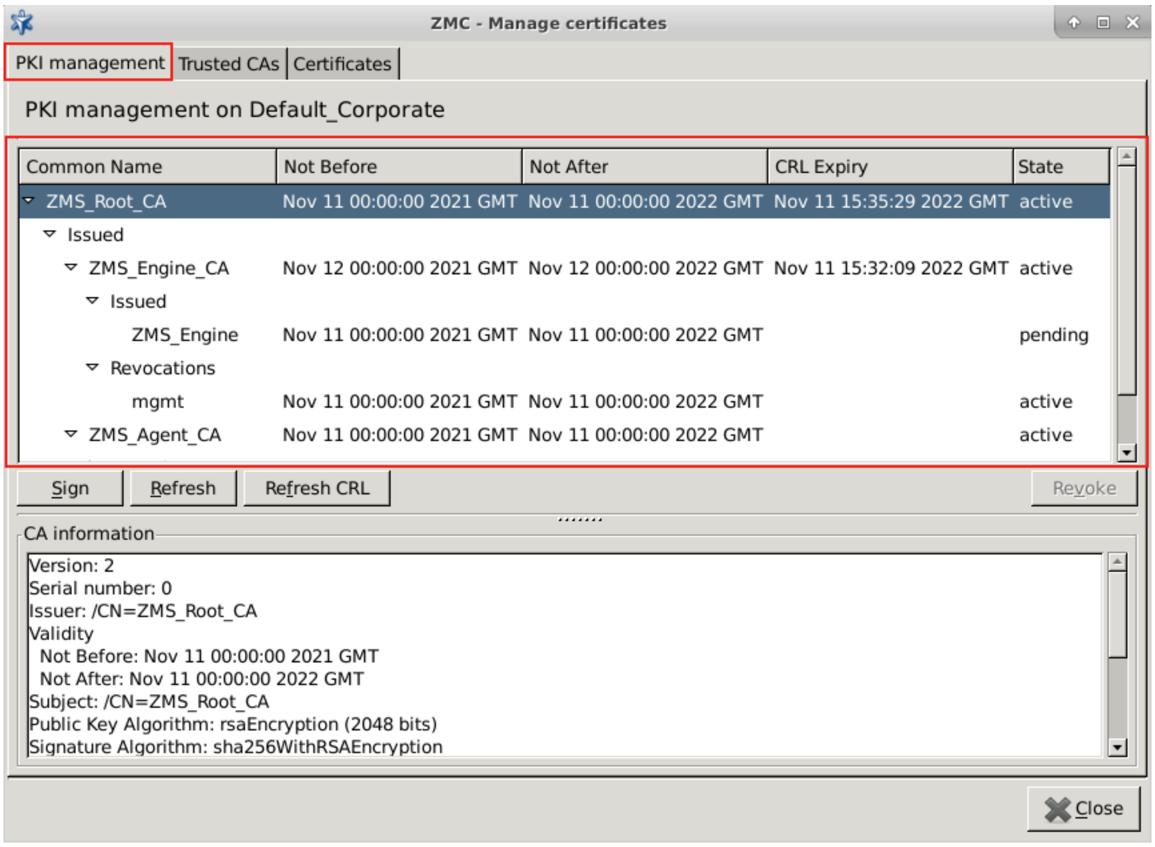
The following data is displayed:

- the distinguished name of the CA issuing the certificate
- the subject of the certificate
- the validity period of the certificate
- the information on the algorithm used to generate the keys, including the length of the key
- any X.509 extensions used in the certificate

Note
The X.509 standard for certificates supports the use of various extensions, for example, to specify for what purposes the certificate can be used, and so on. For details on the possible extensions, see *Appendix C, Further readings (p. 485)*.

11.3.6. PKI management

A tree-like navigation window displays the managed internal CAs. On a newly installed system only local CAs created by default are available. Expired certificates are shown in red.


ZMC - Manage certificates
PKI management
Trusted CAs
Certificates
PKI management on Default_Corporate

Common Name	Not Before	Not After	CRL Expiry	State
ZMS_Root_CA	Nov 11 00:00:00 2021 GMT	Nov 11 00:00:00 2022 GMT	Nov 11 15:35:29 2022 GMT	active
Issued				
ZMS_Engine_CA	Nov 12 00:00:00 2021 GMT	Nov 12 00:00:00 2022 GMT	Nov 11 15:32:09 2022 GMT	active
Issued				
ZMS_Engine	Nov 11 00:00:00 2021 GMT	Nov 11 00:00:00 2022 GMT		pending
Revocations				
mgmt	Nov 11 00:00:00 2021 GMT	Nov 11 00:00:00 2022 GMT		active
ZMS_Agent_CA	Nov 11 00:00:00 2021 GMT	Nov 11 00:00:00 2022 GMT		active

```
Version: 2
Serial number: 0
Issuer: /CN=ZMS_Root_CA
Validity
 Not Before: Nov 11 00:00:00 2021 GMT
 Not After: Nov 11 00:00:00 2022 GMT
Subject: /CN=ZMS_Root_CA
Public Key Algorithm: rsaEncryption (2048 bits)
Signature Algorithm: sha256WithRSAEncryption
```

Figure 11.7. The PKI management navigation window

The internal CAs have small arrows that can be used to display the certificates issued and revoked by the CA.

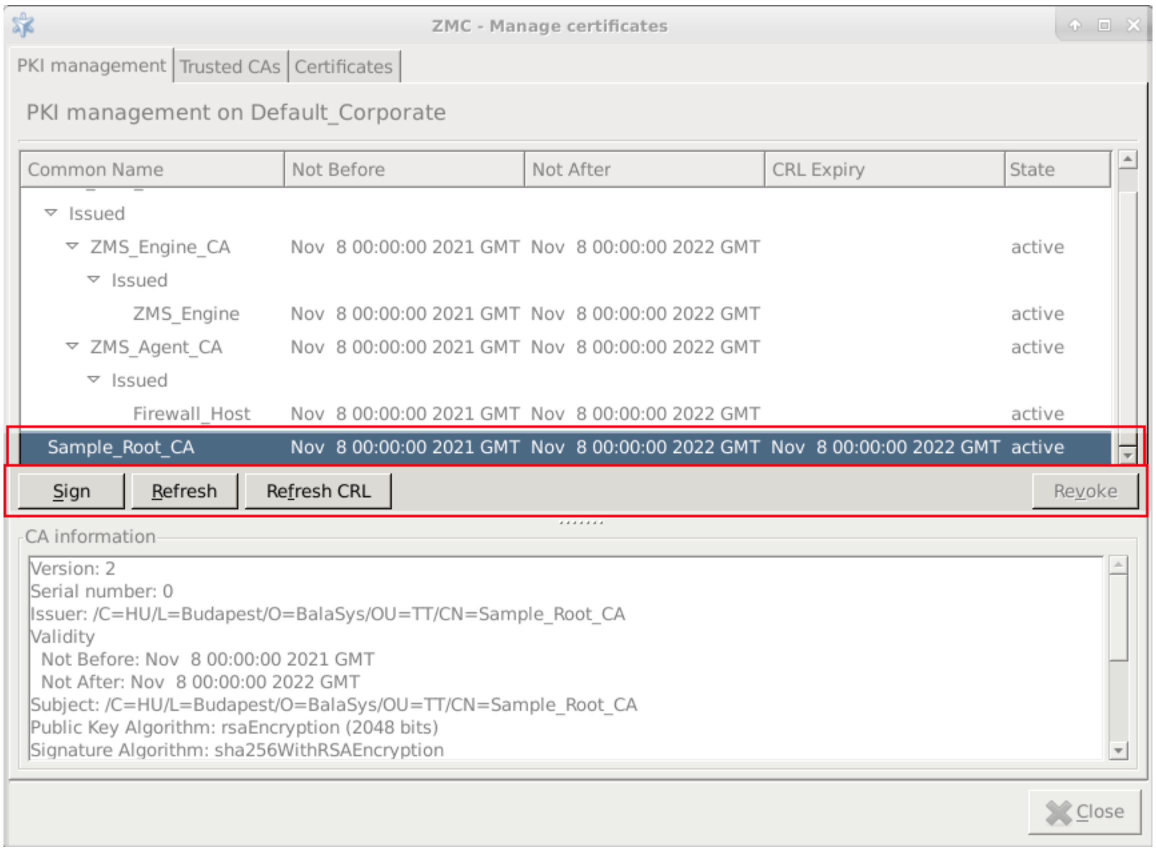
For a given certificate, the following information is displayed:

- the common name of the certificate
- the validity (not before and not after)
- the state, whether the certificate is active (a) or pending (p)
  A certificate becomes pending if the certificate of the CA issuing it (or the certificate of a CA higher in the CA chain) is refreshed. A certificate has to be refreshed if its validity period has expired, even if its private key has not changed. This is because the hash of the refreshed certificate is different from the old one.

Warning
When the certificate of a CA is refreshed, all certificates issued by the CA have to be refreshed (reissued) as well. If the CA has issued certificates for sub-CAs, then also the certificates issued by these subCAs have to be refreshed.

11.3.6.1. The command bar of PKI management

The *Command bar* of the *PKI management* window contains the different commands that can be issued for the certificate or the CA selected.



PKI management on Default_Corporate

Common Name	Not Before	Not After	CRL Expiry	State
Issued				
ZMS_Engine_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
Issued				
ZMS_Engine	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
Issued				
Firewall_Host	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
Sample_Root_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	Nov 8 00:00:00 2022 GMT	active

CA information

```
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=Sample_Root_CA
Validity
 Not Before: Nov  8 00:00:00 2021 GMT
 Not After: Nov  8 00:00:00 2022 GMT
Subject: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=Sample_Root_CA
Public Key Algorithm: rsaEncryption (2048 bits)
Signature Algorithm: sha256WithRSAEncryption
```

Figure 11.8. PKI management commands

The available commands are:

- **Sign**: This action is available only for internal CAs, used to sign certificate signing requests (CSRs). After clicking on it, a list of unsigned CSRs is displayed. The list shows the distinguished names of the CSRs. Parameters for the certificate to be signed can be overridden here (period of validity, X.509 extensions, and so on).

  Note
  It is possible to multi-select a number of certificates for this activity, that is to sign multiple internal CAs or CSRs at once.

- **Refresh**: This command can be used to refresh certificates, that is, to renew them by extending their validity period if expired, or also to create new keys to the certificate. Key generation is only performed if the **Regenerate private key** checkbox is selected.

Tip
It is recommended to regenerate the keys as well when refreshing a certificate for any reason.

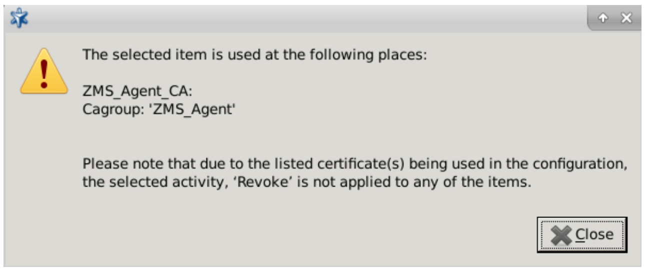
- **Refresh CRL**: It is available only for CAs. The CRL of the CA is valid until the time specified. The refreshed CRL will only be used on the managed hosts after distribution. ZMS distributes certificate entities, that is, when distributing certificates the corresponding CRLs are automatically distributed as well.
- **Revoke**: It is available only for certificates signed by an internal CA. It marks the certificate as invalid and adds it to the CRL of the CA. CA certificates can also be revoked this way.

Note
Self-signed certificates (that is, certificates of local root CAs) cannot be revoked.

Note
It is possible to multi-select a number of certificates for the *Revoke* activity. However, if the Issuer of the selected certificates is not the same, the *Revoke* button will not be active.

Note
If any certificate selected for *Revoke* is in use in the current configuration, a warning will be displayed to inform the administrator. It is important that in case a certificate is in use, it cannot be revoked. If the certificate in use is part of a multiple selection of certificates for the *Revoke* activity, none of the selected certificates will be revoked.

If any of the certificates selected for *Revoke* is used in the configuration, a similar warning is displayed:


The selected item is used at the following places:

ZMS_Agent_CA:
Cagroup: 'ZMS_Agent'

Please note that due to the listed certificate(s) being used in the configuration, the selected activity, 'Revoke' is not applied to any of the items.

Figure 11.9. Certificate used in the configuration warning

The table below briefly summarizes the CAs created and used by default in Zorp.

Name of the CA	Purpose
`ZMS_Root_CA`	The Root CA of Zorp is used to sign certificates of all other local CAs in Zorp.
`ZMS_Engine_CA`	It signs the certificate of the ZMS engine.

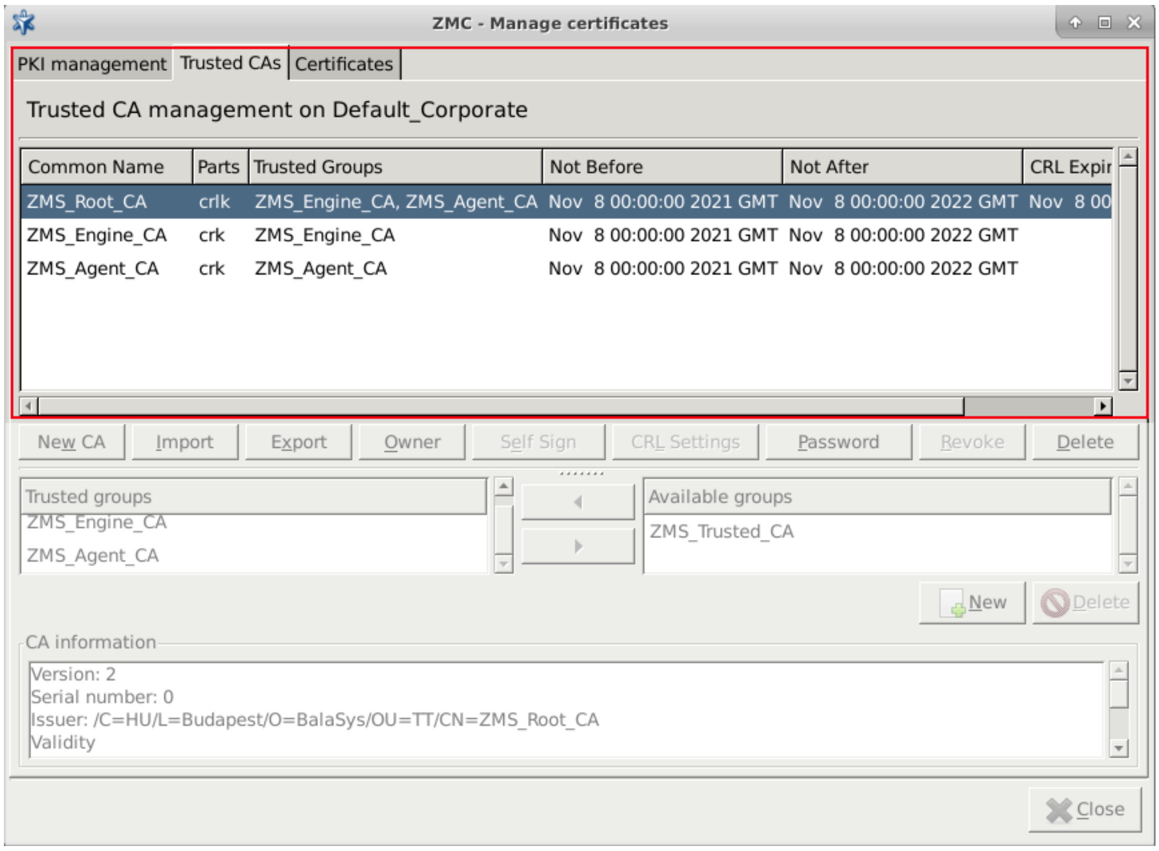
Name of the CA	Purpose
`ZMS_Agent_CA`	It signs the certificates of the transfer agents.

Table 11.1. Default CAs and their purpose

For details on configuring agent and engine certificates, please refer to *Chapter 13, Advanced ZMS and Agent configuration (p. 323)*.

11.3.7. Trusted CAs

This menu item is for managing certificate authorities. The upper section of the panel displays the list of available CAs, both internal and external. Apart from creating the default internal CAs, a number of trustworthy and external certificates (for example, VeriSign, NetLock) are imported as well.



ZMC - Manage certificates

PKI management | Trusted CAs | Certificates

Trusted CA management on Default_Corporate

Common Name	Parts	Trusted Groups	Not Before	Not After	CRL Expir
ZMS_Root_CA	crlk	ZMS_Engine_CA, ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	Nov 8 00
ZMS_Engine_CA	crk	ZMS_Engine_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	
ZMS_Agent_CA	crk	ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	

Trusted groups: ZMS_Engine_CA, ZMS_Agent_CA

Available groups: ZMS_Trusted_CA

CA information

```
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Root_CA
Validity
```

Figure 11.10. Trusted CAs

The following information is displayed on each CA:

- **Common Name**: It displays the common name of the CA.
- **Parts**: It denotes the components of the certificate entity available for the CA.
  - *c*: It stands for *certificate*. Usually this is the only part available for external CAs.
  - *k*: It denotes the private key of the certificate.
  - *r*: It refers to the certificate signing request (CSR).

  - *l*: It is the CRL of the CA.

  An internal CA is fully functional if all of its parts are available (CRL is optional).

- **Trusted groups**: It denotes the name(s) of the trusted groups that the CA is member of.
- **Not before/not after**: It defines the validity of the CAs' certificate.
- **CRL expiry**: It defines the date until the CRL of the given CA is valid. If this field is empty, no CRL has been released by the CA so far.

11.3.7.1. The command bar of Trusted CAs

The command bar contains various operations that can be performed with the CAs. Some of them require a CA to be selected from the information window, in this case the given operation will be performed on the CA selected. Note, that for some of the activities multi-select option is available for performing mass activity. These possibilities are described in details at each activity.

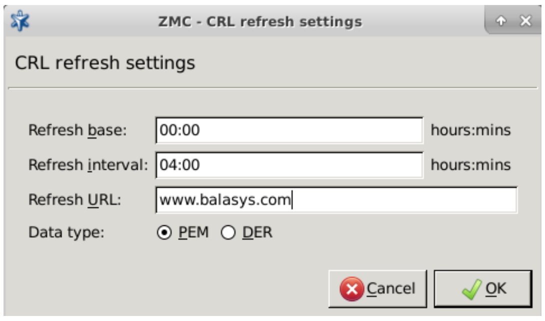
- **New CA**: Create a new local Certificate Authority. For details, see *Procedure 11.3.7.2, Creating a new CA (p. 270)*. As CAs require unique names, they can only be created one by one.
- **Import**: Import a CA certificate from a PEM, DER, or PKCS12 formatted file. It is possible to import only one CA at once. The *Import into selected object* can only be selected if at the time of the **Import** only one line is selected in the Trusted CA list.
- **Export**: Export the certificate of the selected CA into a file in PEM, DER, or PKCS12 format. The PKCS12 format is only available for internal CAs.
- **Owner**: A CA available on a site, can be made available on all sites managed by ZMS, by clicking this button and checking in the **Available on all sites** checkbox. Making a CA certificate available on all sites cannot be reversed, that is, once a CA has been made available on all sites, later it cannot be limited to a single site. This has the same effect as checking in the corresponding checkbox when creating a new CA.

Warning
This operation cannot be reversed or undone.

- **Self sign**: Self-sign the certificate signing request (CSR) of the selected local CA. Only certificates not yet signed by a CA can be self-signed. This activity can be implemented one by one on the items, no multi-selection is possible.

Note
Local root CAs can be created by self-singing a so far unsigned CSR of a Trusted CA.

- **CRL settings**: Set the parameters for refreshing the CRL of the selected external CA. Note, that only single-selection is possible.



Refresh base: 00:00 hours:mins
Refresh interval: 04:00 hours:mins
Refresh URL: www.balasys.com
Data type: [✓ PEM] DER

Figure 11.11. CRL settings

The following parameters can be set:

- **Refresh base**: It defines at what time the retrieval of the CRL shall be started.
- **Refresh interval**: It defines how often the CRL shall be retrieved.

  By setting the **Refresh base** to 00:00 and the **Refresh interval** to 04:00, the CRL will be downloaded every four hours, starting from midnight.

- **Refresh URL**: The location of the CRL can be retrieved with this parameter setting. The CRL can be downloaded through HTTP.

  Note
  It is very important to set the refresh URL option, otherwise the validity of the certificates issued by the CA cannot be reliably verified. The CRL shall be downloaded and automatically distributed regularly.

- **Data type**: It defines the format of the CRL to be downloaded (PEM or DER).

- **Password**: Change the password of the selected local CA or it is possible to define a password here if it has not been configured yet.
- **Revoke**: Revoke the certificate of the selected local CA that was signed by another local CA. Self-signed CA certificates cannot be revoked this way. For details, see *Procedure 11.3.8.3, Revoking a certificate (p. 280)*.

  Note
  It is possible to multi-select a number of certificates for the *Revoke* activity. However, if the Issuer of the selected certificates is not the same, the *Revoke* button will not be active.

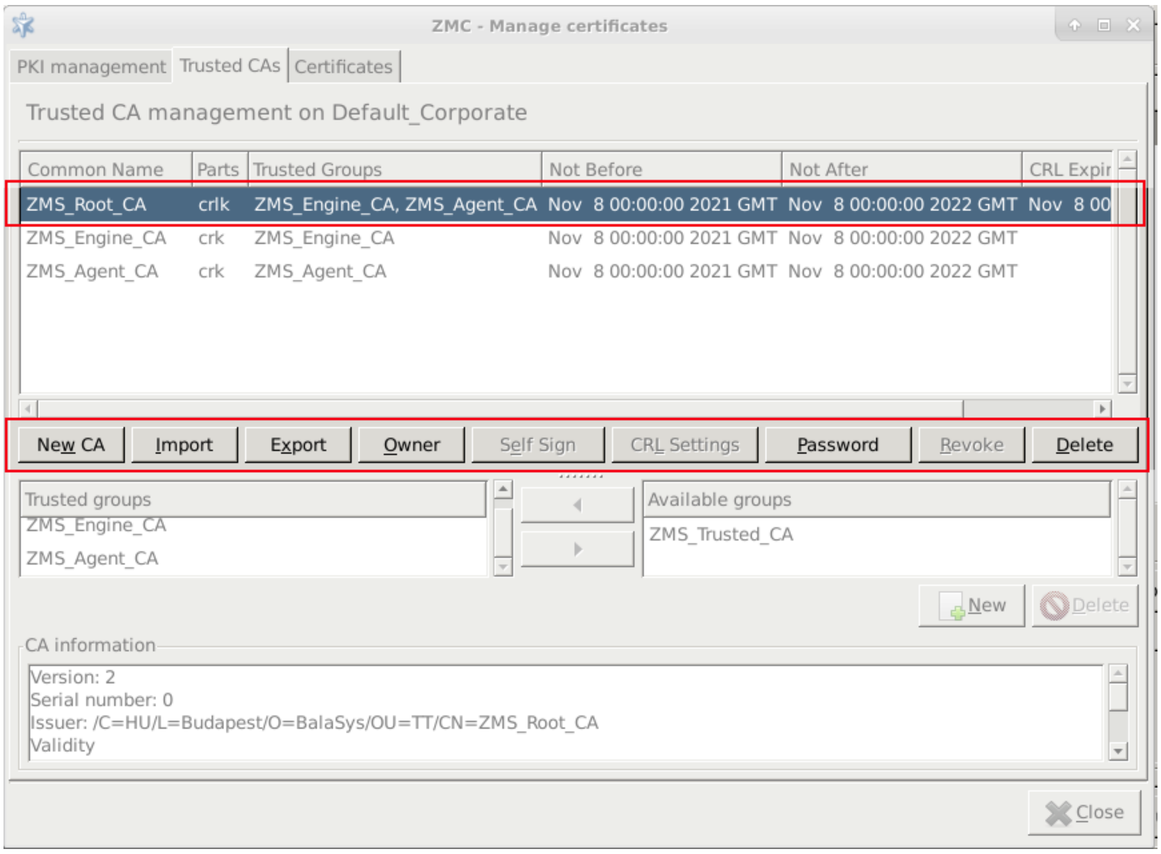
  Note
  If the certificate(s) selected for *Revoke* is in use in the current configuration, a warning will be displayed to inform the administrator. It is important that in case a certificate is in use, it cannot be revoked. If the certificate in use is part of a multiple selection of certificates for the *Revoke* activity, none of the selected certificates will be revoked.

- **Delete**: Delete the selected certificate. For details, see *Procedure 11.3.8.4, Deleting certificates (p. 282)*.

Note
It is possible to multi-select a number of certificates for the *Delete* activity. If the certificate(s) selected for *Delete* is in use in the current configuration, a warning will be displayed to inform the administrator. It is important that in case a certificate is in use, it cannot be deleted. If the certificate in use is part of a multiple selection of certificates for the *Delete* activity, none of the selected certificates will be deleted.

11.3.7.2. Procedure – Creating a new CA

Step 1. Navigate to the **Trusted CAs** tab of the **PKI/Edit certificates menu**, and click on **New CA**.



Trusted CA management on Default_Corporate

Common Name	Parts	Trusted Groups	Not Before	Not After	CRL Expir
ZMS_Root_CA	crlk	ZMS_Engine_CA, ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	Nov 8 00
ZMS_Engine_CA	crk	ZMS_Engine_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	
ZMS_Agent_CA	crk	ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	

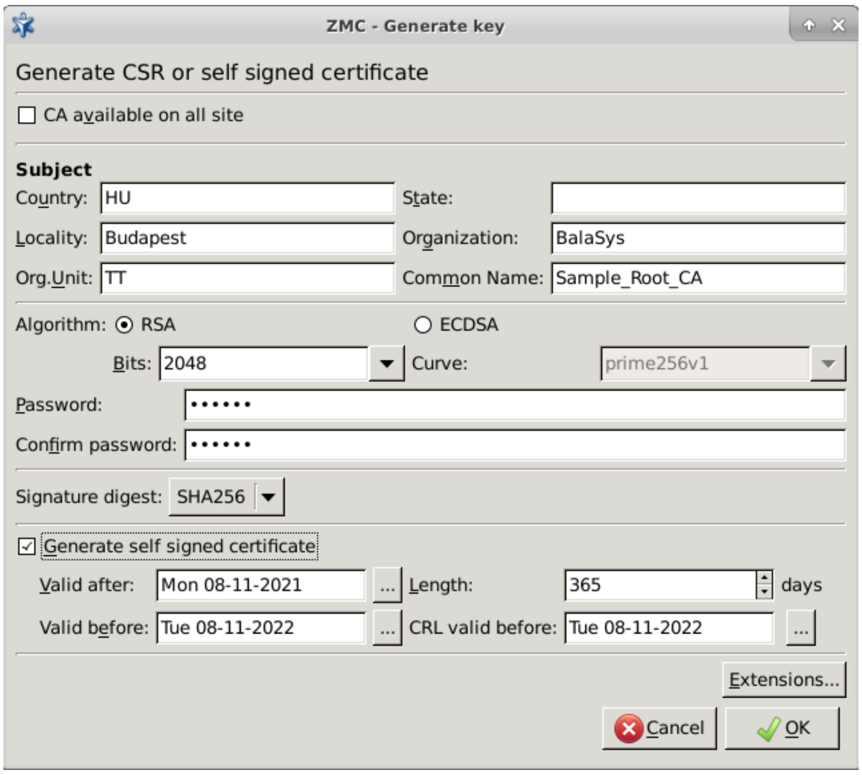
Trusted groups
ZMS_Engine_CA
ZMS_Agent_CA

Available groups
ZMS_Trusted_CA

CA information
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Root_CA
Validity

Figure 11.12. The Trusted CAs command bar

Step 2. Enter the required parameters for the subject of the new CA's certificate. It is required that the CA has a unique **Common Name**, but is is also helpful if the **Common Name** is descriptive as well, as it helps to remember the CA's function later.


ZMC - Generate key

Generate CSR or self signed certificate

☐ CA available on all site

Subject

Country:	HU	State:	
Locality:	Budapest	Organization:	BalaSys
Org.Unit:	TT	Common Name:	Sample_Root_CA

Algorithm: ● RSA ○ ECDSA

Bits: 2048 Curve: prime256v1

Password: ••••••

Confirm password: ••••••

Signature digest: SHA256

☑ Generate self signed certificate

Valid after: Mon 08-11-2021 Length: 365 days

Valid before: Tue 08-11-2022 CRL valid before: Tue 08-11-2022

Figure 11.13. Creating a new CA

Step 3. Select the encryption algorithm and key length to be used.

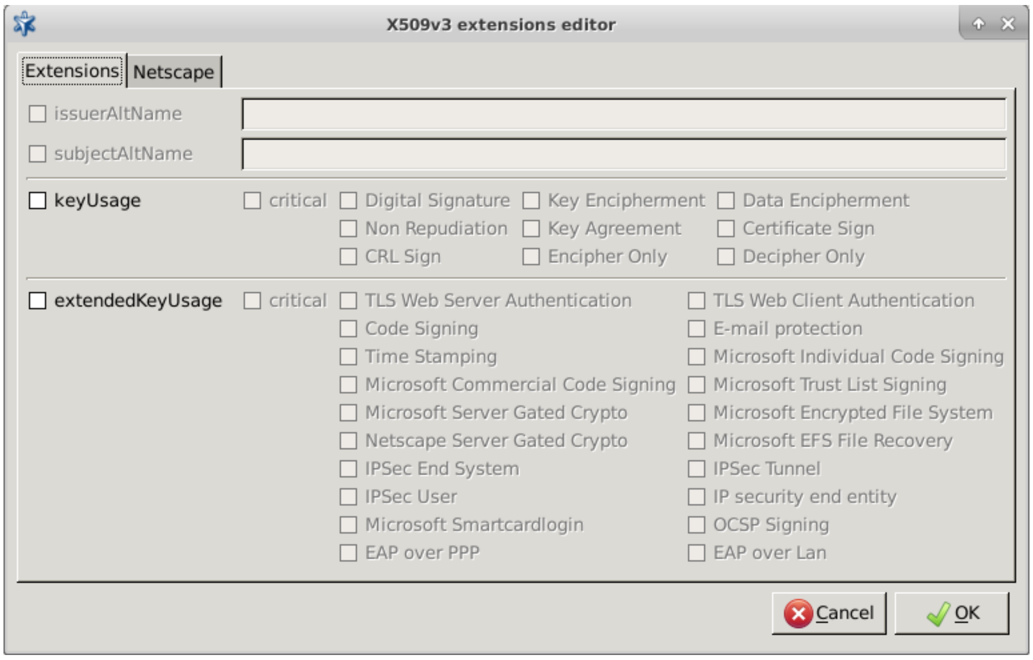
Tip
The key of the CA certificate shall be longer than the ones that will be issued by the CA, for example, if the CA is used to sign certificates having 1024 bit keys, the key of the CA certificate shall be at least 2048 bit long.

Step 4. Select the signature digest (hash) method to be used.

Tip
Use of the SHA1 algorithm is recommended, as it is considered to be more secure and not significantly more computation intensive.

Step 5. Provide a password to protect the private key of the CA. This is required so that only authorized users can sign certificates.

Step 6. Click on **Extensions ...**, and specify for which purposes the certificate will be used.



- [ ] issuerAltName
- [ ] subjectAltName

☐ keyUsage	☐ critical	☐ Digital Signature	☐ Key Encipherment	☐ Data Encipherment
		☐ Non Repudiation	☐ Key Agreement	☐ Certificate Sign
		☐ CRL Sign	☐ Encipher Only	☐ Decipher Only

☐ extendedKeyUsage	☐ critical	☐ TLS Web Server Authentication	☐ TLS Web Client Authentication
		☐ Code Signing	☐ E-mail protection
		☐ Time Stamping	☐ Microsoft Individual Code Signing
		☐ Microsoft Commercial Code Signing	☐ Microsoft Trust List Signing
		☐ Microsoft Server Gated Crypto	☐ Microsoft Encrypted File System
		☐ Netscape Server Gated Crypto	☐ Microsoft EFS File Recovery
		☐ IPSec End System	☐ IPSec Tunnel
		☐ IPSec User	☐ IP security end entity
		☐ Microsoft Smartcardlogin	☐ OCSP Signing
		☐ EAP over PPP	☐ EAP over Lan

Figure 11.14. Specifying extensions

Note
The use of extensions is optional.

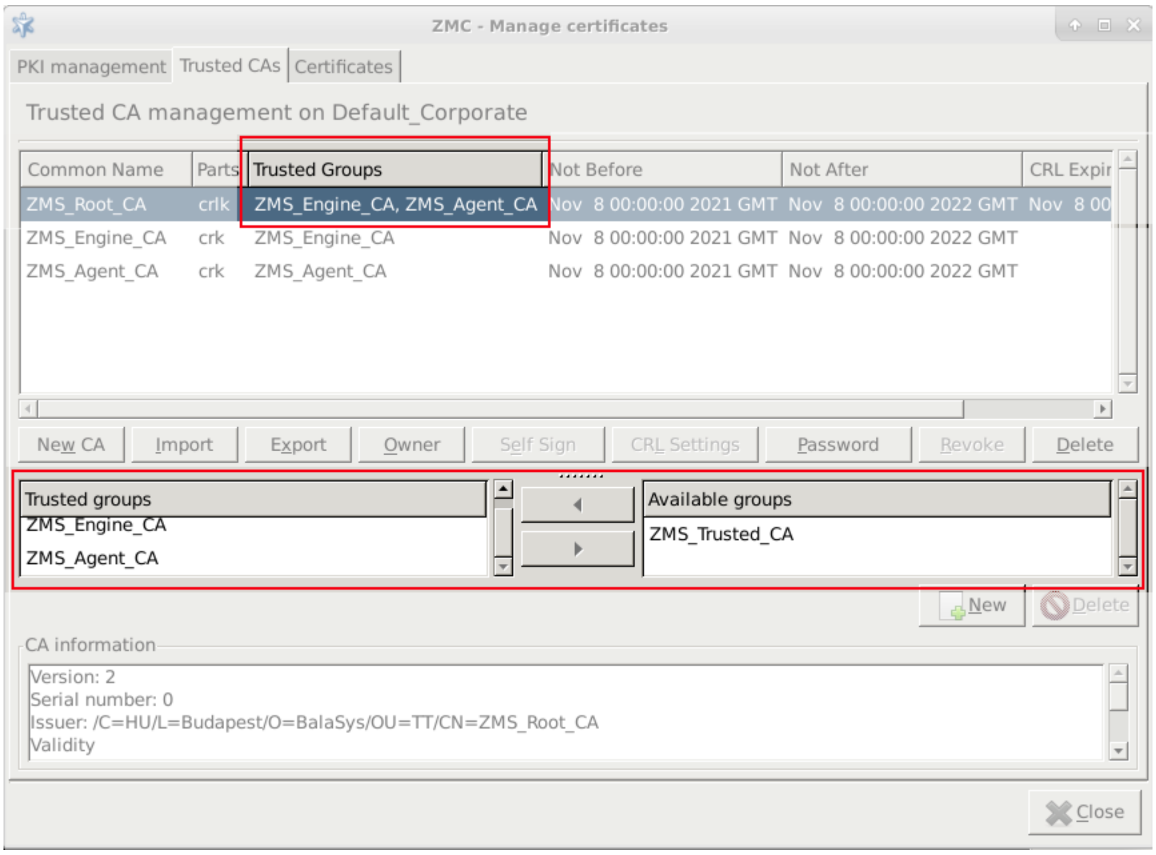
Step 7. When creating a local root CA, check the **Generate self-signed certificate** checkbox and specify the validity period of the certificate.

Tip
If the CA is to be available on every site managed, do not forget to check in the appropriate checkbox when creating the New CA.

Warning
A CA available on a site, can be made available on all sites managed by ZMS, by checking in the **Available on all sites** checkbox. Making a CA certificate available on all sites cannot be reversed, that is, once a CA has been made available on all sites, later it cannot be limited to a single site.

11.3.7.3. Managing trusted groups

The **Available groups** are displayed on the right side of the panel, while **Trusted groups**, listing the groups that the selected CA is member of, are displayed on the left. Adding or removing a CA to a group can be performed by selecting the CA for configuration, selecting or multi-selecting the groups that are wished to be moved and using the arrow-shaped icons in the middle.



Trusted CA management on Default_Corporate

Common Name	Parts	Trusted Groups	Not Before	Not After	CRL Expir
ZMS_Root_CA	crlk	ZMS_Engine_CA, ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	Nov 8 00
ZMS_Engine_CA	crk	ZMS_Engine_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	
ZMS_Agent_CA	crk	ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	

Trusted groups
ZMS_Engine_CA
ZMS_Agent_CA

Available groups
ZMS_Trusted_CA

CA information

```
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Root_CA
Validity
```

Figure 11.15. Trusted CA groups

11.3.7.4. Procedure – Signing CA certificates with external CAs

If you want to use an external CA to sign the certificate of a local CA, complete the following steps.

Step 1. Generate a private-public keypair and an associated certificate signing request (CSR) using the **Generate** button of the **Certificates** tab.

Step 2. Export this CSR into a file using the **Export** button.

Step 3. Have the CSR signed.

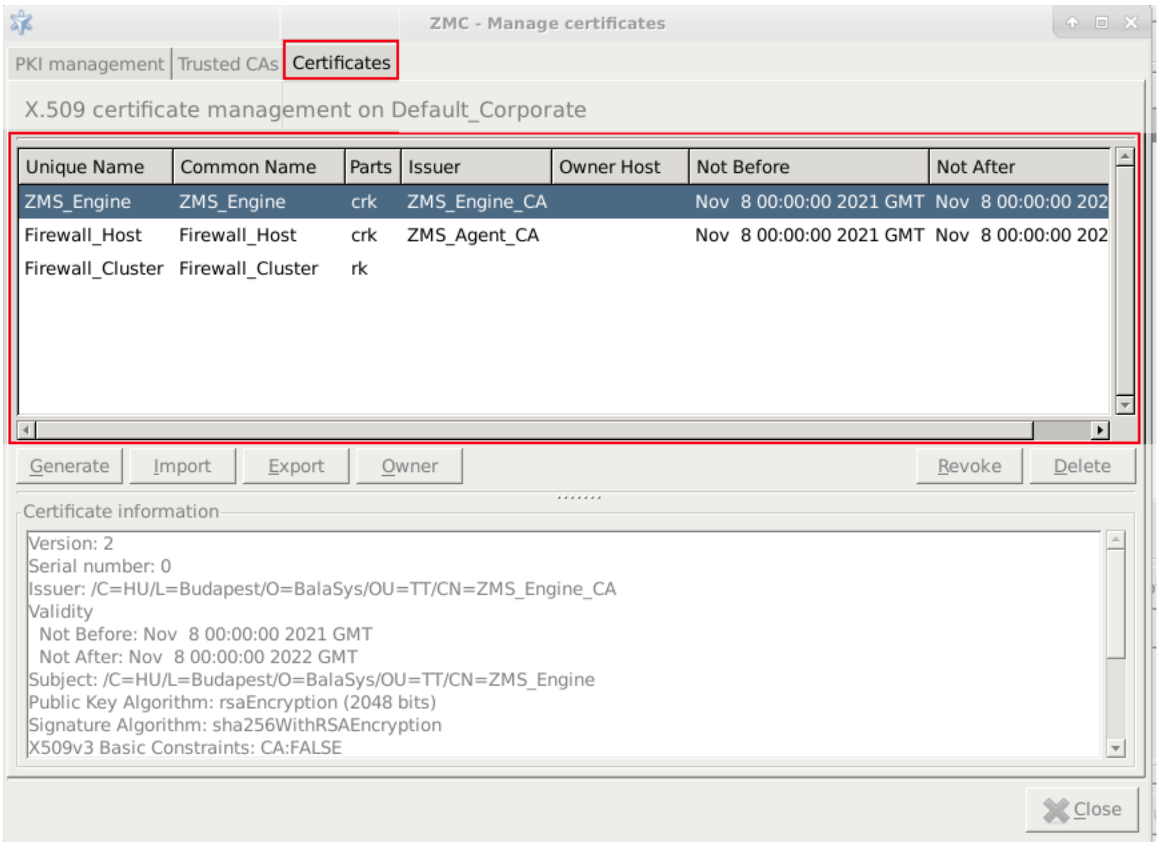
Step 4. If the CA approves your identity and signs the certificate, **Import** it to the PKI system of ZMS.

Note
Make sure the appropriate entity is selected (that is, the signed certificate to the proper CSR is imported) and the **Import into selected object** option is checked in.

Step 5. The certificate entity can now be distributed and used on your machines.

11.3.8. Managing certificates

All non-CA certificates available on the selected site can be managed here. It is also possible to import and export certificates.



X.509 certificate management on Default_Corporate

Unique Name	Common Name	Parts	Issuer	Owner Host	Not Before	Not After
ZMS_Engine	ZMS_Engine	crk	ZMS_Engine_CA		Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 202
Firewall_Host	Firewall_Host	crk	ZMS_Agent_CA		Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 202
Firewall_Cluster	Firewall_Cluster	rk				

Certificate information

```
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Engine_CA
Validity
 Not Before: Nov  8 00:00:00 2021 GMT
 Not After: Nov  8 00:00:00 2022 GMT
Subject: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Engine
Public Key Algorithm: rsaEncryption (2048 bits)
Signature Algorithm: sha256WithRSAEncryption
X509v3 Basic Constraints: CA:FALSE
```

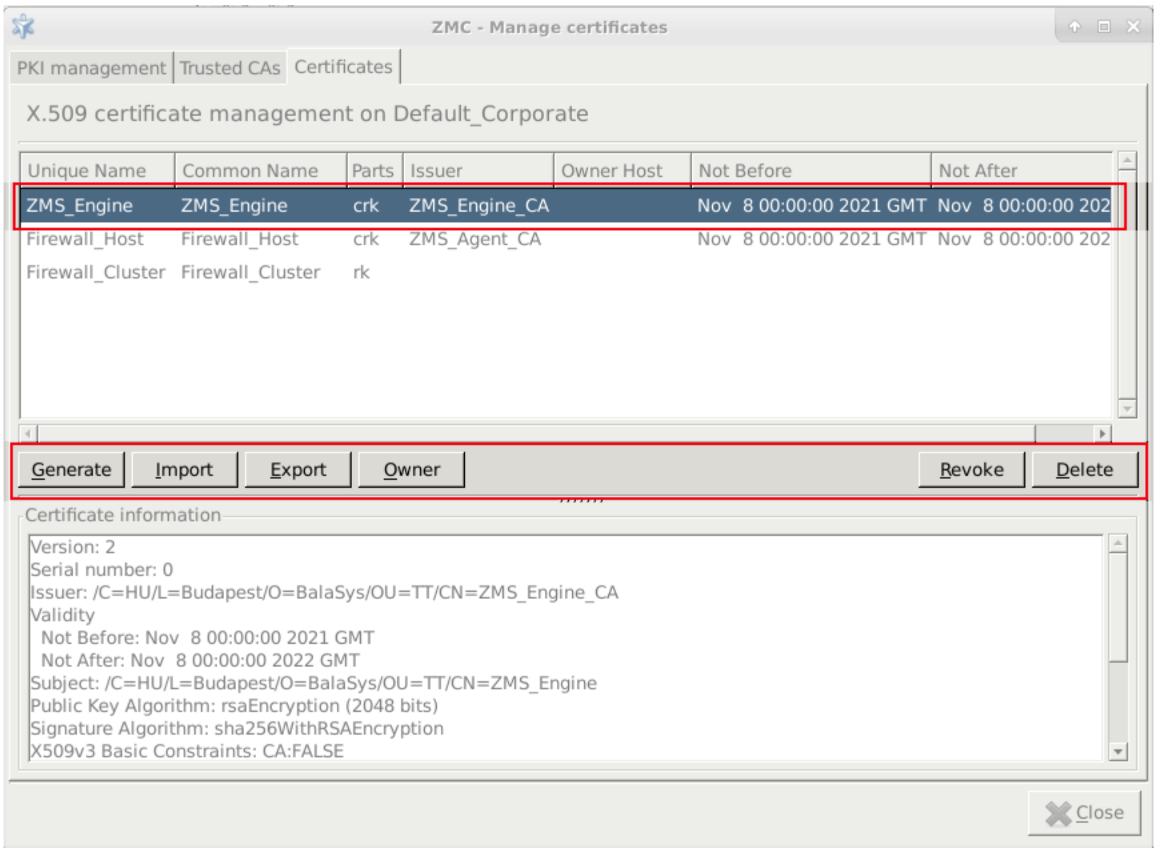
Figure 11.16. Certificates

The upper section of the panel displays the list of available certificates, along with the following information on each:

- **Unique name**: It is the unique name of the certificate entity.
- **Common name**: It is the common name of the certificate.
- **Parts**: It displays the available parts from the certificate entity:
  - c: For external certificates usually only their certificate (c) is available.
  - k: For internal certificates their private key (k) can also be available.
  - r: For internal certificates the certificate signing request (CSR) can also be available.
- **Issuer**: This is the CA that has signed the certificate. This field is empty if the CSR is not yet signed.
- **Owner host**: It determines which host can use the private key.
- **Not before/Not after**: The certificate's period of validity.

11.3.8.1. The Certificates command bar

The command bar contains buttons that can be used to perform various operations on the certificates available on the site. Some of them require a certificate to be selected from the information window, in this case the given operation will be performed on the selected certificate.



X.509 certificate management on Default_Corporate

Unique Name	Common Name	Parts	Issuer	Owner Host	Not Before	Not After
ZMS_Engine	ZMS_Engine	crk	ZMS_Engine_CA		Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 202
Firewall_Host	Firewall_Host	crk	ZMS_Agent_CA		Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 202
Firewall_Cluster	Firewall_Cluster	rk				

Certificate information

```
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Engine_CA
Validity
  Not Before: Nov  8 00:00:00 2021 GMT
  Not After: Nov  8 00:00:00 2022 GMT
Subject: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=ZMS_Engine
Public Key Algorithm: rsaEncryption (2048 bits)
Signature Algorithm: sha256WithRSAEncryption
X509v3 Basic Constraints: CA:FALSE
```

Figure 11.17. The Certificates command bar

- **Generate**: Generate a new certificate signing request (CSR).
- **Import**: Import a certificate (or a part of it) from a PEM, DER, or PKCS12 formatted file. It is possible to import only one certificate at once. The *Import into selected object* can only be selected if at the time of the **Import** only one line is selected in the *Certificates* list.
- **Export**: Export the selected certificate (or a part of it) into file in PEM, DER, or PKCS12 format. In case of exporting one certificate the name of the exported file has to be provided. Note, that the certificate must have a **Common Name**. In case a private key is also exported, a password can also be defined for it. However, it is also possible to select multiple certificates for **Export**. In that case the certificates will be exported to a selected folder into files named after their unique names. In case private keys are also exported, the passwords belonging to these private keys will be generated into the same folder, named after the unique names of the certificates, with .txt file extension format.
- **Owner**: The owner host of the certificate can be specified here. Also, the certificate available on a site, can be made available on all sites managed by ZMS, by checking in the **Available on all sites** checkbox. This function is reversible though and the owner host can also be changed later as well.
- **Revoke**: Revoke the selected certificate. This operation requires the password of the issuer CA.

Note
It is possible to multi-select a number of certificates for the *Revoke* activity. However, if the Issuer of the selected certificates is not the same, the *Revoke* button will not be active.

Note
If the certificate(s) selected for *Revoke* is in use in the current configuration, a warning will be displayed to inform the administrator. It is important that in case a certificate is in use, it cannot be revoked. If the certificate in use is part of a multiple selection of certificates for the *Revoke* activity, none of the selected certificates will be revoked.

- **Delete**: Delete the selected certificate. For details, see *Procedure 11.3.8.4, Deleting certificates (p. 282)*.

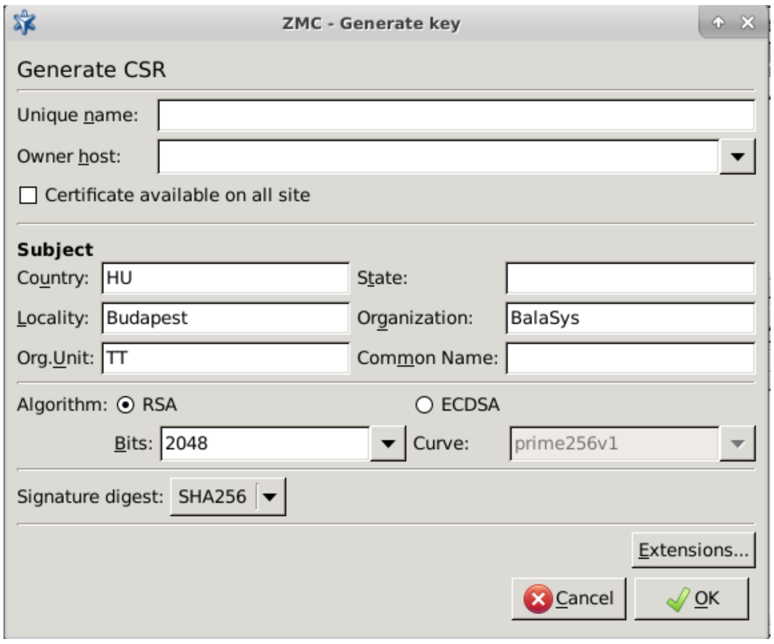
Note
It is possible to multi-select a number of certificates for the *Delete* activity. However, if the certificate(s) selected for *Delete* is in use in the current configuration, a warning will be displayed to inform the administrator that the group of certificates selected for deletion cannot be deleted. Note, that even if only one certificate is in use among the selected elements, none of the certificates will be deleted.

11.3.8.2. Procedure – Creating certificates

To create a certificate, complete the following steps.

Step 1. Select **PKI > Edit Certificates** from the menu and click **Certificates**.

Step 2. Click **Generate**, and fill the Generate CSR form.


ZMC - Generate key

Generate CSR

Unique name:

Owner host:

☐ Certificate available on all site

Subject

Country: HU
State:
Locality: Budapest
Organization: BalaSys
Org.Unit: TT
Common Name:

Algorithm: [✓ RSA] ECDSA

Bits: 2048
Curve: prime256v1

Signature digest: SHA256

Figure 11.18. Creating a certificate

Step a. Enter a **Unique Name** that will identify the object containing the certificate and the key in ZMS. Note, that in case after filling in the **Unique Name** field, the **Enter** button is used, the value of the **Unique Name** field is also added to the **Common Name** field.

Step b. Select the host from the combobox, who will be the owner of the certificate.

Step c. If you want the certificate to be available on every site that is managed in ZMS, select **Certificate available on all sites**.

Step d. Fill the Subject section of the request as appropriate. Into the **Country** field, enter only a two-letter ID (for example, US). Enter a name for the certificate into the **Common name** field. Note that in case the fields have been filled in at **Site preferences**, those values will automatically be offered here.

Step e. Select the length of the key (1024, 2048, or 4096 bit).

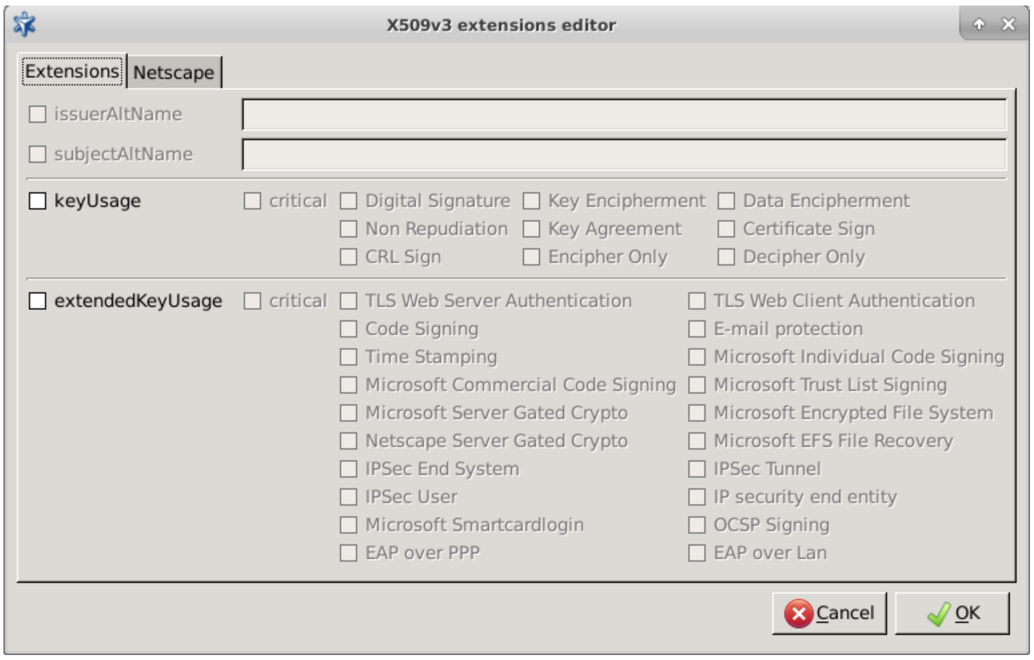
Note
Longer keys are more secure, but the time needed to process key signing and verification operations (required for using encrypted connections) increases exponentially with the length of the key used. By default, 2048 bit is used.

ZMC 7 can create only RSA keys, generating DSA keys is not supported.

Warning
If the certificates/keys have to be used on machines running older versions of the Windows operating system, using only 1024 bit long keys might be required, since these Windows versions typically do not support longer keys.

Step f. Select the method (SHA256 or SHA512) to be used for generating the Signature digest (hash).

Step g. By clicking on **Extensions ...**, the various purposes of the certificate can be specified. For details on X.509v3 extensions, see *Appendix C, Further readings (p. 485)*.



☐ issuerAltName
☐ subjectAltName

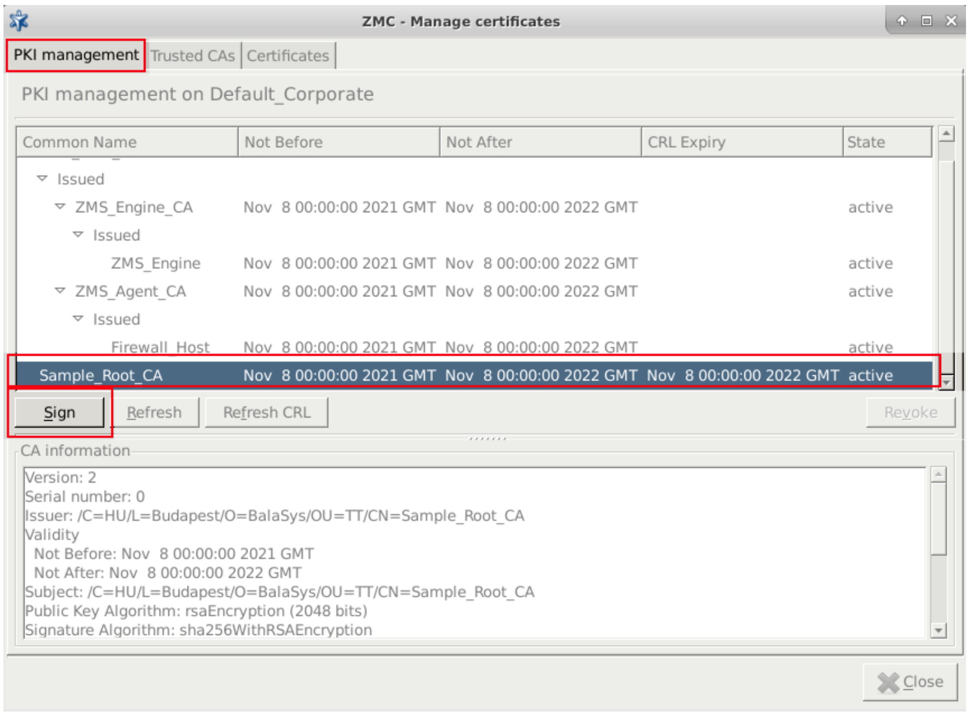
☐ keyUsage ☐ critical
☐ Digital Signature ☐ Key Encipherment ☐ Data Encipherment
☐ Non Repudiation ☐ Key Agreement ☐ Certificate Sign
☐ CRL Sign ☐ Encipher Only ☐ Decipher Only

☐ extendedKeyUsage ☐ critical
☐ TLS Web Server Authentication ☐ TLS Web Client Authentication
☐ Code Signing ☐ E-mail protection
☐ Time Stamping ☐ Microsoft Individual Code Signing
☐ Microsoft Commercial Code Signing ☐ Microsoft Trust List Signing
☐ Microsoft Server Gated Crypto ☐ Microsoft Encrypted File System
☐ Netscape Server Gated Crypto ☐ Microsoft EFS File Recovery
☐ IPSec End System ☐ IPSec Tunnel
☐ IPSec User ☐ IP security end entity
☐ Microsoft Smartcardlogin ☐ OCSP Signing
☐ EAP over PPP ☐ EAP over Lan

Figure 11.19. Specifying extensions

Step h. After specifying all the required options, click **OK**.

Step 3. Navigate to the **PKI management** tab, and in the navigation window select the local CA to be used to sign the request (for example, `ZMS_Agent_CA` for transfer agents, and so on).



PKI management on Default_Corporate

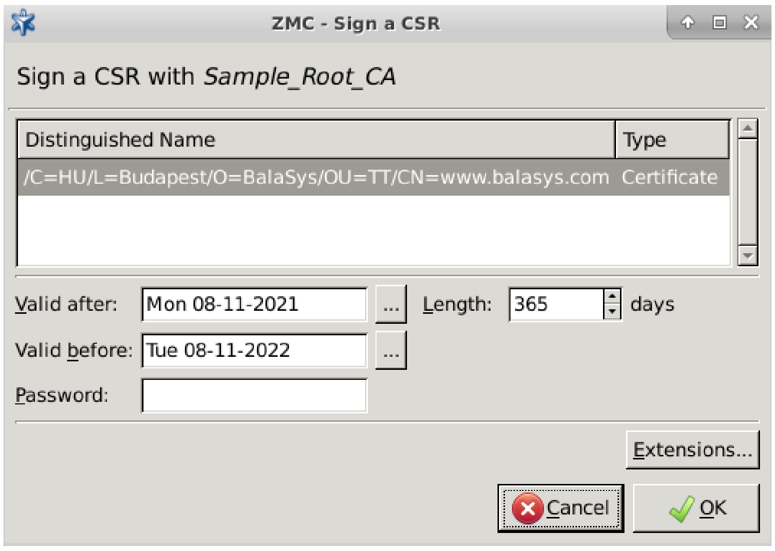
Common Name	Not Before	Not After	CRL Expiry	State
Issued				
ZMS_Engine_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
Issued				
ZMS_Engine	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
ZMS_Agent_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
Issued				
Firewall_Host	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT		active
Sample_Root_CA	Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2022 GMT	Nov 8 00:00:00 2022 GMT	active

CA information

```
Version: 2
Serial number: 0
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=Sample_Root_CA
Validity
  Not Before: Nov  8 00:00:00 2021 GMT
  Not After: Nov  8 00:00:00 2022 GMT
Subject: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=Sample_Root_CA
Public Key Algorithm: rsaEncryption (2048 bits)
Signature Algorithm: sha256WithRSAEncryption
```

Figure 11.20. Signing a certificate

Step 4. Click on **Sign**. A window will be displayed listing the submitted but not yet signed certificate signing requests (CSRs). Note, that it is possible to use multi-select here. The list displays the distinguished name of the CSRs, this includes the various Subject fields (Country, locality, common name, and so on) specified when generating the request.



Sign a CSR with *Sample_Root_CA*

Distinguished Name	Type
/C=HU/L=Budapest/O=BalaSys/OU=TT/CN=www.balasys.com	Certificate

Valid after: Mon 08-11-2021 ... Length: 365 days

Valid before: Tue 08-11-2022 ...

Password:

Figure 11.21. Selecting the certificate to be signed

Step 5. Set the validity period (**Valid after**/**Valid before** dates) of the certificate. A pop-up calendar is available through the **...** button. Alternatively, after setting the Valid after date, the **Length** field can be used to specify the length of the validity in days, automatically updating the **Valid before** field.

Step 6. By clicking on **Extensions ...**, various X.509 extensions can be specified. These extensions can be used to ensure in filters that only certificates used for their intended purpose are accepted.

Note
Note that although similar configration details can be defined when creating a certificate - and also different settings can be defined for each certificate, the settings defined here will overwrite any other configuration settings and only these settings will be applicable.

Step 7. Enter the password of the CA required for issuing new certificates, and click **OK**.

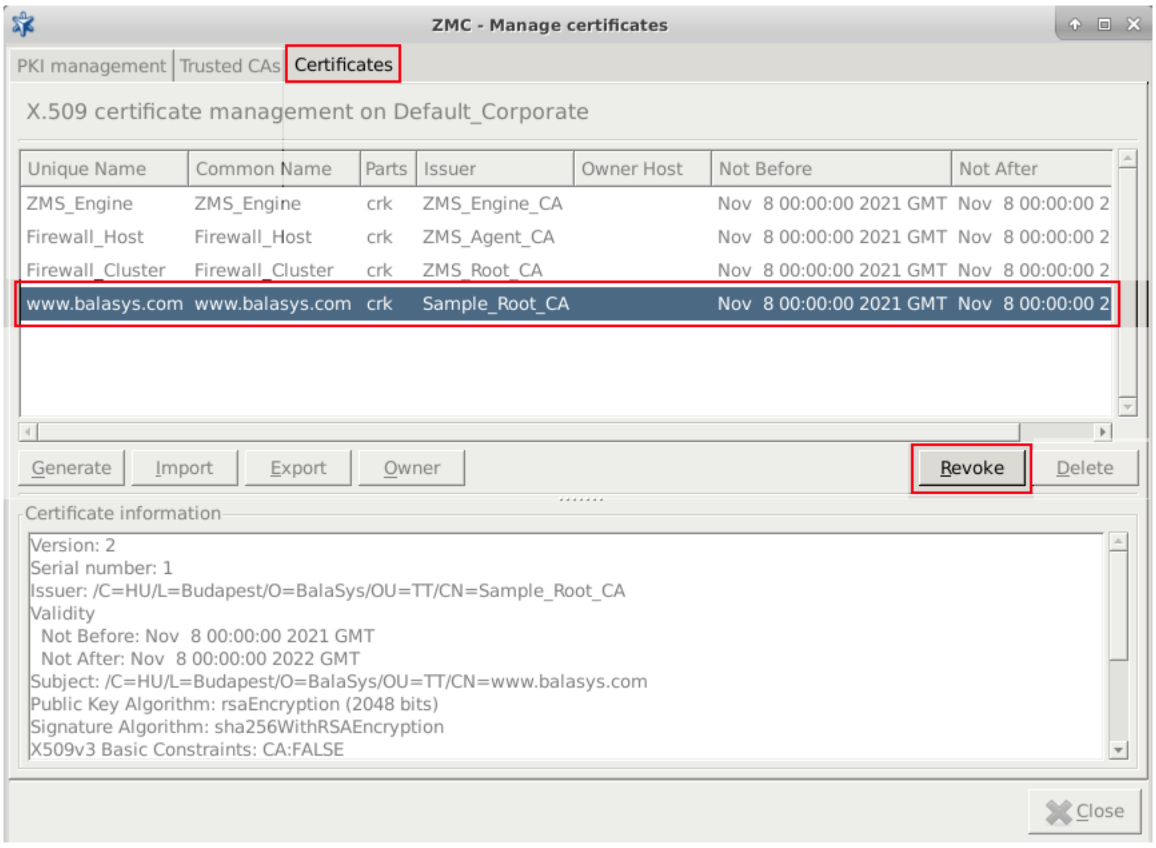
11.3.8.3. Procedure – Revoking a certificate

To revoke a certificate, complete the following steps.

Step 1. Select the certificate to be revoked.
Note, that it is possible to multi-select a number of certificates for the *Revoke* activity. However, if the certificate has no Issuer, the *Revoke* button will not be active.

Note
It is possible to multi-select a number of certificates for the **Revoke** activity. However, if the Issuer of the selected certificates is not the same, the Revoke button will not be active.

Note
Note that if the certificate(s) selected for *Revoke* is in use in the current configuration, a warning will be displayed to inform the administrator. It is important that in case a certificate is in use, it cannot be revoked. If the certificate in use is part of a multiple selection of certificates for the *Revoke* activity, none of the selected certificates will be revoked.



X.509 certificate management on Default_Corporate

Unique Name	Common Name	Parts	Issuer	Owner Host	Not Before	Not After
ZMS_Engine	ZMS_Engine	crk	ZMS_Engine_CA		Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2
Firewall_Host	Firewall_Host	crk	ZMS_Agent_CA		Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2
Firewall_Cluster	Firewall_Cluster	crk	ZMS_Root_CA		Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2
www.balasys.com	www.balasys.com	crk	Sample_Root_CA		Nov 8 00:00:00 2021 GMT	Nov 8 00:00:00 2

Certificate information

```
Version: 2
Serial number: 1
Issuer: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=Sample_Root_CA
Validity
  Not Before: Nov  8 00:00:00 2021 GMT
  Not After: Nov  8 00:00:00 2022 GMT
Subject: /C=HU/L=Budapest/O=BalaSys/OU=TT/CN=www.balasys.com
Public Key Algorithm: rsaEncryption (2048 bits)
Signature Algorithm: sha256WithRSAEncryption
X509v3 Basic Constraints: CA:FALSE
```

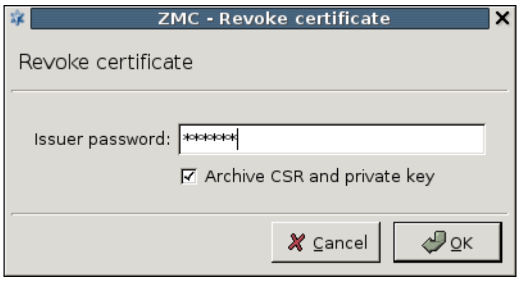
Figure 11.22. Revoking certificates

Step 2. For general certificates, click on **Revoke** either on the **PKI management** or the **Certificates** tab. CA certificates can be revoked from either the **PKI management** or the **Trusted CAs** tab.

Note
Only certificates signed by local CAs can be revoked.

Self-signed CA certificates cannot be revoked.

Step 3. Enter the password of the issuer CA. If the private key associated to the certificate is to be revoked as well, check the **Archive CSR and private key** checkbox. Click **OK**.



[✓ Archive CSR and private key]

Figure 11.23. Revoking the private key

Tip
If the private key of a certificate has been compromised, the private key should be revoked along with the certificate. Generally it is recommended to generate new keys each time a certificate is refreshed.

Step 4. Following the **Revoke** of the certificate, the certificate will disappear from the lists of certificates on the **Certificates** tab, and will only appear on the **PKI management** tab, in the **Revocations** list of its CA.

Step 5. The CRL of the issuer CA is refreshed automatically.

Step 6. The revocation will be effective on the Zorp hosts only when their CRL information is updated from ZMS. If ZMS is not configured to perform distribution automatically (or the update should be made available immediately), it can be performed manually through the **PKI/Distribute Certificates** menu item.

11.3.8.4. Procedure – Deleting certificates

To delete a certificate from the ZMS PKI, complete the following steps.

Step 1. Select the certificate you want to delete on the **Certificates** tab, and click **Delete**.

Note
It is possible to multi-select a number of certificates for the *Delete* activity. However, if the certificate(s) selected for *Delete* is in use in the current configuration, a warning will be displayed to inform the administrator that the group of CAs selected for deletion cannot be deleted. Note, that even if only one CA is in use among the selected elements, none of the CAs will be deleted.

Step 2. From the main menu, select **PKI** > **Distribute Certificates**.

11.3.8.5. Procedure – Exporting certificates

To export a certificate from the ZMS PKI, complete the following steps.

Step 1. Select the certificate to be exported on the **Certificates** tab.

Note
When only one certificate is exported, the name of the exported certificate file has to be defined during the export. When a number of certificates are exported at once with the help of multiple selection, the exported certificates will be named based

on their unique names. If by the export of multiple certificates, private keys are also exported, passwords must also be generated. The passwords of the private keys will be saved in .txt files format into the same directory the certificates are exported to and will be named based on the unique name of the corresponding certificate they belong to.

Step 2. Click on **Export**.

Step 3. Select the directory to save the file(s) to. Specify the filename in case of single certificate export (as in case of exporting multiple certificates, the names are automatically created), and click **OK**.

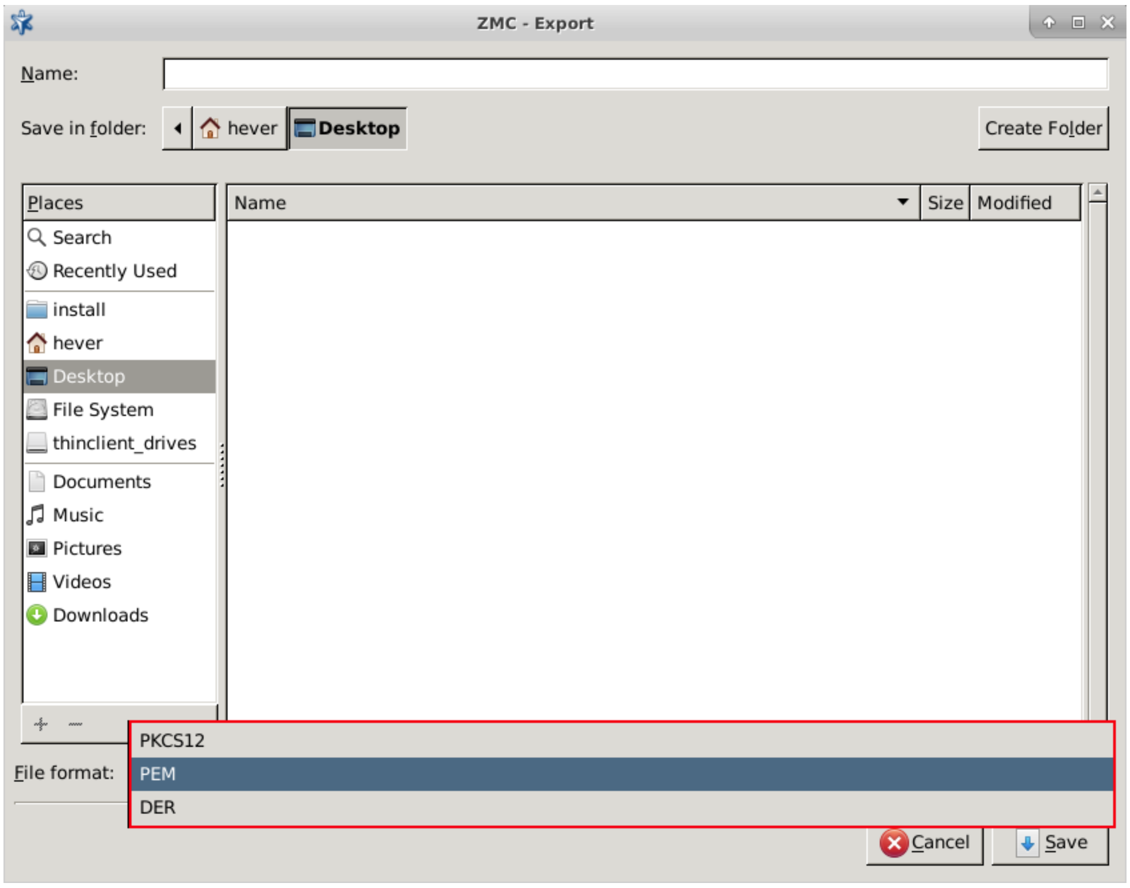


Figure 11.24. Exporting certificates

Note
File extension is NOT added automatically to the filename.

The file will be saved to the local machine, that is, the one that is running ZMC.

Step 4. Depending on the file format to be used, the part(s) to be saved can be specified. Naturally, only the parts that are available can be selected (for example, only the CSR or the key if the certificate has not been signed yet).

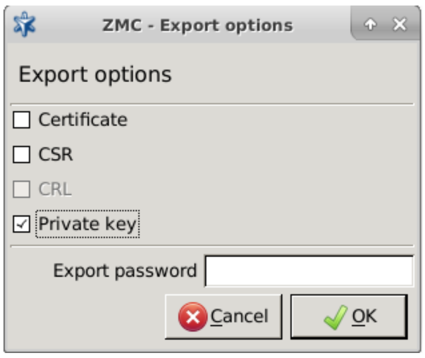


Figure 11.25. Selecting certificate components to export

Step 5. If the private key is exported as well, in case of exporting one certificate, the key can be password-protected by specifying an **Export password**. In case more than one certificate is exported, and private keys are also selected for export, the passwords belonging to the private keys will be automatically generated into the same folders, where the certificates are exported to and will be saved into .txt files. The .txt files will be named after the unique name of the certificate they belong to.

11.3.8.6. Procedure – Importing certificates

Step 1. Click on **Import** on the **Trusted CAs** tab for importing a CA certificate, or on the **Certificates** tab for normal certificates.
It is possible to import only one CA or certificate at once, and consequently, the *Import into selected object* can only be selected if at the time of the **Import** only one line is selected in the Trusted CA list or in the Certificates list.

> **Note**
> If the parts contained in the file are to be imported for being added to an existing certificate entity, select the given certificate entity before clicking on **Import**. This function is useful for importing certificates signed by external CAs.

Step 2. Specify the file format to be used, and select the file to be imported.

Step 3. Select the part(s) to be imported.

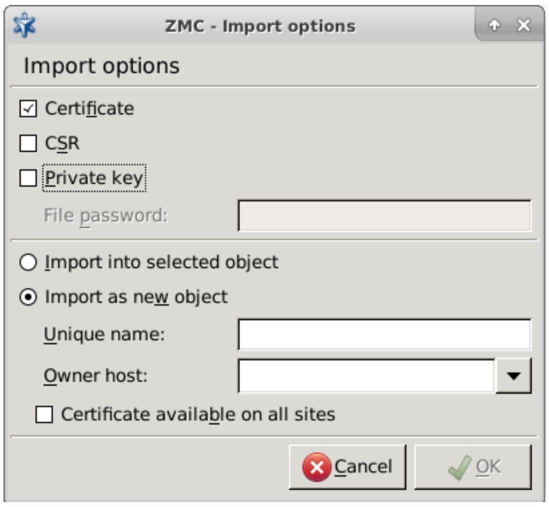


Figure 11.26. Importing certificates

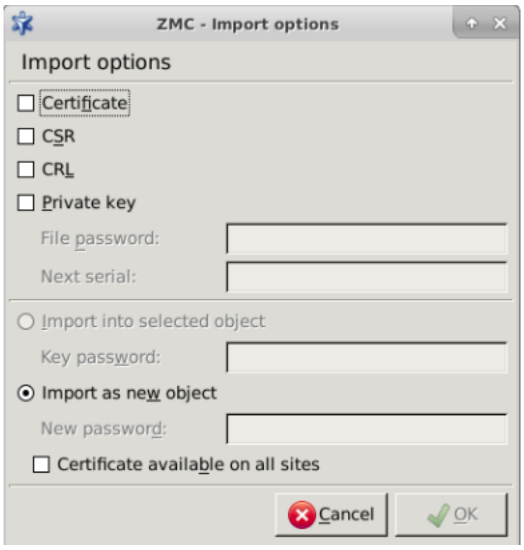


Figure 11.27. Importing CA certificates

Step 4. There are two ways to handle the data imported from the file: creating a new entity or appending them to an existing one.
Creating a new entity: Select the **Import as new object** radio button, enter a Unique name and you can also select the Owner host of the object if needed. This method is useful especially for importing the certificates of external CAs.

Import parts to an existing certificate: It is possible to import the part(s) contained in the file into an existing certificate entity (that is, the one that was selected before clicking on the **Import** button). This method should be used when importing your certificates that were signed by an external CA, so the certificate is imported to the entity containing the private key and the CSR. Select the **Import into selected object** radio button.

Step 5. Enter the **Export password** if the private key is imported and the key has been password-protected.

Step 6. Check in the **Certificate available on all sites** checkbox if needed.

11.3.8.7. Procedure – Signing your certificates with external CAs

If you have an external CA to sign your certificates and you want to manage these certificates in Zorp, complete the following steps.

Tip
The Import and Export operations provide a convenient way to handle certificates signed by external CAs. For details, see *Procedure 11.3.8.6, Importing certificates (p. 284)* and *Procedure 11.3.8.5, Exporting certificates (p. 282)*.

Step 1. Generate a private-public keypair and an associated CSR using the **Generate** button of the **Certificates** tab.

Step 2. Export this CSR into a file using the **Export** button.

Step 3. Have the CSR signed.

Step 4. If the CA approves your identity and signs the certificate, **Import** it to the PKI system of ZMS.

Note
Make sure the appropriate entity is selected (that is, the signed certificate to the proper CSR is imported) and the **Import into selected object** option is checked in.

Step 5. The certificate can now be distributed and used on your machines.

11.3.8.8. Procedure – Monitoring licenses and certificates

Purpose:

The ZMS and Zorp hosts monitor the validity of product licenses and certificates, and automatically send alert e-mails if any of them will expire soon. By default, the host sends e-mail alert to the administrator e-mail address specified during the installation, 14 days before the expiry.

The validity of the available product licenses (ZMS, Zorp, ZCV, ZAS, NOD32) is checked once every day on each host that is managed from ZMS. The validity of CA certificates and certificates is checked only on the ZMS host.

To configure the details of the certificate monitoring, complete the following steps.

Steps:

Step 1. Add a new **Text editor** component to the host to edit the `/etc/zmsagent/expiration.conf` file. For details on editing a file using the **Text editor** component, see *Procedure 8.1.1, Configure services with the Text editor plugin (p. 213)*.

Step 2. Configure the address of the mailserver, and the e-mail address of the recipients as needed. For details on the available parameters, see the manual page of expiration.conf.

Step 3. Commit and upload your changes.

Step 4. Execute the `/etc/cron.daily/expiration_check.py --test-mail` command to send a test e-mail.

Chapter 12. Clusters and high availability

12.1. Introduction to clustering

A cluster is a group of computers dedicated to performing the same task. These computers (referred to as nodes of the cluster) use the same (or very similar) configuration files (policies, iptables, and so on). The goal of clustering in general is to integrate the resources of two or more devices (that could otherwise function separately) together for backup, high availability or load sharing purposes. In other words, clusters are computer systems in which more than one computer shares the tasks or the load in the network. A Zorp cluster usually consists of a group of firewall hosts that maintain the same overall security policy and share the same configuration settings.

Basically there are two types of clusters. In a failover cluster if a machine breaks down, a spare computer is started immediately to ensure that the service provided by the computers is continuously available (see *Section 12.2.1, Fail-Over clusters (p. 288)*). Load balancing clusters are used when the traffic generated by the provided service is more what a single computer can handle (see *Section 12.2.2*, *Load balance clusters (p. 290)*).

Clustering provides the following advantages:

- ensures continuous service and decreased downtime,
- contributes to High Availability (HA),
- assists to satisfy service level agreements, and
- improves load balance in the system.

The following terms will be frequently used in this chapter:

Host	A single computer offering services to the clients.
Node	A single computer belonging to a cluster, offering services to the clients exactly with the same functionality as the other nodes in the cluster.
Cluster	A (logical and physical) group of computers offering services to the clients. Clusters are made up of nodes. In ZMS, the nodes of a cluster are handled together: from the administration point of view a cluster behaves similarly to a single host.

12.2. Clustering solutions

12.2.1. Fail-Over clusters

The aim of failover clustering is to ensure that the service is accessible even if one of the servers breaks down (for example, because of a hardware error). In failover clusters only one of the nodes is functioning and carries the whole traffic, the other(s) only monitor the active node. In case of system failure resulting in a loss of service, the service is started on the other node in the system. In other words, if the active component dies the other node takes over all the services.

Note
The service fails over to the other component only in case of hardware failure, if you stop Zorp no backup mechanism is initiated automatically.

Monitoring between the cluster nodes is realized with the help of Heartbeat messages. For more information, see *Section 12.5, Heartbeat (p. 299)*.

The transfer of services can be realized using one of the following methods:

- Transferring the Service IP address
- Transferring IP and MAC address
- Sending RIP messages

12.2.1.1. Service IP transferring

In this case the servers use a virtual (alias) IP address (called Service IP), clients access the service provided by the servers by targeting this Service IP. The Service IP is carried by the active node only. If the node providing the service (that is, the master node) fails, the Service IP is taken over by the slave node. As all clients send requests towards the Service IP, they are not aware of which device that address belongs to, and do not notice any difference when a takeover occurs.

When the cluster relocates the Service IP to the other node it sends a *gratuitous ARP request* message to the whole network informing the clients that the Service IP belongs now to a different node. As a result, the clients flush their *ARP* cache and record the new *ARP* address of the Service IP. (A gratuitous ARP request is an AddressResolutionProtocol request packet where the source and destination IP are both set to the IP of the machine issuing the packet and the destination MAC is the broadcast address ff:ff:ff:ff:ff:ff. Ordinarily, no reply packet will occur.)

Service IP takeover is the most frequently used takeover method for Zorp clusters.

Note
Some clients may not take over the new Service IP address until the next automatic ARP cache flush, which causes certain delay in Service IP transfers in the system. The problem is that the ARP cache is refreshed relatively rarely, and it is not possible to notify the clients to update their ARP cache.

12.2.1.2. IP with MAC address takeover

In some systems, usually in large networks it is disadvantageous to modify IP address – Media Access Control (MAC) pairs, because certain routers do not refresh their ARP cache, causing problems in the network traffic. In this case the failover functionality is realized by taking over the Media Access Control (MAC) address.

In such systems, all nodes use the same fix IP and hardware MAC address in the network and the nodes are differentiated by the state of the servicing interface. The master (active) node has the interface in up state while the slave nodes' interfaces are kept down. If the service fails over to the other node, the interfaces get into up state. Client requests are serviced by the node having the interface in up state.

Transferring MAC address is beneficial if the resources need to be relocated very quickly.

Warning
Multiple interfaces with the same IP or MAC address connecting to a network as a result of a failed takeover can destabilize the network. Consequently, it is important to monitor the takeover process and to completely remove (for example, power off) the failed server from the network. This can be accomplished by using a STONITH device. See *Section 12.5, Heartbeat (p. 299)* for details.

12.2.1.3. Sending RIP messages

In this case no Service IP is used. All nodes have their own IP addresses and the routing information is sent through Routing Information Protocol (RIP) messages using different metrics.

Note
RIP metrics are ranging from 1 to 6. You have to define the metrics for each node according to your network environment.

Routers in the network select the destination components based on these metrics. They send the traffic to the node with lower metrics value. If a node fails, it is sufficient to remove it from the network and traffic is transferred through the other nodes without any further interaction.

Note
The router mediating the client requests towards the firewall has to support RIP message transfer. Desktop clients and common server machines usually do not support RIP messages.

Zorp uses the Sendrip software for this purpose.

12.2.2. Load balance clusters

In load balance configurations, all nodes of a cluster provide services simultaneously to distribute system load and enhance the overall quality of service. Clients access the service by targeting a single domain name, without knowing how many servers provide the actual service.

The amount of traffic handled by one node is determined by some logic. Currently Zorp does not provide any built-in tool for defining such criteria, therefore an external device has to be used. This can be either the DNS server, or a dedicated load balancer.

12.2.2.1. DNS load balancing

DNS load balancing is based on the native behavior of name resolution, that is when the DNS server resolves a domain name into more than one IP address the client chooses one IP from the answers using the round robin algorithm. Though for the decision the client disregards the actual load of the server, the solution results in balanced load in the system. In this case the firewalls offer a non-transparent service, because the client targets the firewall itself.

12.2.2.2. Load balancing with external devices

It is possible to use load balancer devices to distribute the traffic between the nodes. In this case the balancing method can be configured on the load balancer device. Of course, load balancing solutions also offer a native failover solution. If one node stops working and the load balancing device notices that, it does not direct traffic to that node until it is functioning again.

Load balancer devices offer load balancing only from the point of the client, it has no influence on the proxy at the other side of the firewall — in such case line load balancing must be solved on the firewall. If you need to share a load from several directions (physical networks), separate load balancer devices are needed in each direction.

Note
The firewall has to have a separate load balancer device towards all connected interfaces.

From proxying point of view, all connections, and in case of multi-channel protocols, like FTP, all channels have to go through the same node.

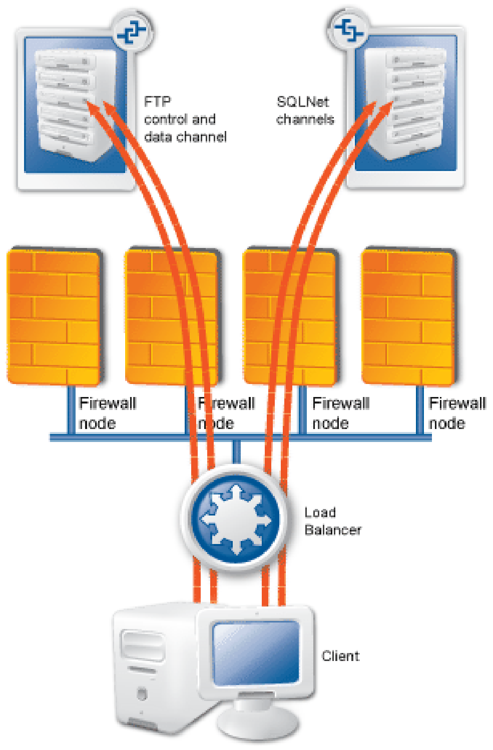


Figure 12.1. Directing related channels to the same node

The third party device added to the system must be able to direct multi-channel protocols through the same node.

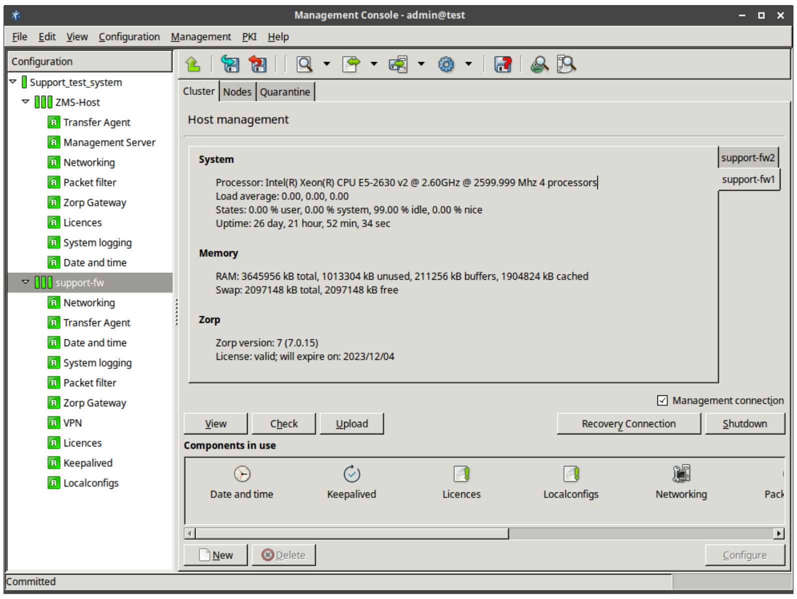
12.2.2.3. Multicast load balancing

A simple load balancing solution is to assign a multicast MAC address to the Service IP. In this case the clients target the Service IP, and a hub or switch before the firewall hosts forwards all requests to the multicast MAC address, resulting in all nodes of the cluster receiving all packets sent to the Service IP. The IP addresses of the clients are distributed between the nodes using some logic (for example, one node serves only clients with odd, the other one clients with even IP addresses), and the packet filter of each node is configured to accept only the packets of the clients they are responsible for.

Note
It is important that if in such a scenario one of the nodes fails, the remaining nodes have to take over the clients served by the failed node. This can be accomplished for example by using **Heartbeat** resources.

12.3. Managing clusters with ZMS

The nodes of a cluster have identical configurations, only a few parameters are different. When configuring clusters, all nodes are configured simultaneously, as if the cluster were a single host.



Host management

System

Processor: Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz @ 2599.999 Mhz 4 processors
Load average: 0.00, 0.00, 0.00
States: 0.00 % user, 0.00 % system, 99.00 % idle, 0.00 % nice
Uptime: 26 day, 21 hour, 52 min, 34 sec

Memory

RAM: 3645956 kB total, 1013304 kB unused, 211256 kB buffers, 1904824 kB cached
Swap: 2097148 kB total, 2097148 kB free

Zorp

Zorp version: 7 (7.0.15)
License: valid; will expire on: 2023/12/04

Figure 12.2. A cluster in ZMC

For each parameter that is different on the nodes of the cluster, links have to be used. It is also possible to link to a property of the cluster, in this case the link will be evaluated to a different value on each node. That way when the configuration is uploaded, each node will receive a configuration file containing the values relevant for the node.

Any parameter can be used as a property; usually parameters like the IP addresses of the interfaces are properties. New properties can be added any time to the cluster, not only during the initial configuration.

Naturally, not all links used in a cluster have to be links to cluster properties, regular links can be used as well. However, keep in mind that links to cluster properties are resolved to the corresponding property of the particular node. For example, a link to the `Hostname` property of a cluster is resolved on each node to the hostname of the node (for example, to *`node_1`* on the first node, and so on).

Note
The PKI of the site considers the cluster to be a single host, there is no difference between the individual nodes.

As a result of using properties, adding new nodes to a cluster is very easy, since only the properties have to be filled with values for the new node.

When uploading configuration changes, or viewing and checking configurations, you can select on which node the operation shall be performed.

Controlling a service (for example, restarting/reloading) is possible on all nodes simultaneously, or only on the nodes specified in the selection window.

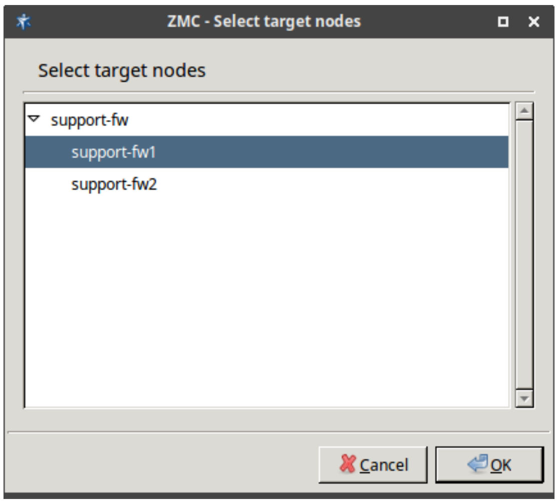


Figure 12.3. Selecting the target node

Status indicator icons on clusters behave identically to hosts, except that a blue led indicates a *partial* status, meaning that the nodes of the cluster are not all in the same state (for example, the configuration was not successfully uploaded to all nodes).

When configuring rules for Zorp clusters, use links to the interfaces. From the clients' point of view this makes no difference, as the clients do not target the IP of the Zorp host.

For non-transparent services, the rule must use the Service IP (that is, a link to the Service IP), because that is where the clients will send their requests to.

12.4. Creating clusters

When configuring a new cluster, there are several distinct steps that have to be completed. An overview of the general procedure is presented below. The main tasks are to create and configure the cluster nodes; to configure Heartbeat (required only for failover clusters and certain load balancing solutions); and finally to create the policies, services on the cluster.

First the new cluster has to be created in ZMC. This can be either a cluster created from scratch, or (optionally) an existing host can be converted into a cluster. In both cases the initial cluster has only a single node, the additional nodes have to be added (and bootstrapped) manually. Bootstrapping a cluster node is very similar to bootstrapping a regular host. It is important to create properties for the parameters that are different on each node (for example, hostname, IP address, and so on) and use links during configuration when referring to these properties.

In case of failover and multicast load balancing clusters, the **Heartbeat** component also has to be installed and configured. For load balancing clusters where the load balancing is performed by an external device (that is, a load balancer, DNS server, and so on), this external device also has to be configured. Configuring **Heartbeat** has two main steps, first the communication between the nodes has to be configured, then the *Heartbeat resources* that are taken over when a node fails have to be created (see *Section 12.5, Heartbeat (p. 299)* for details).

After completing the above procedure, the cluster-specific configuration of the system is finished — later steps can be performed identically to managing the policies of regular hosts.

The individual steps of the above procedure are described in the following sections in detail.

Note
The procedures in the subsequent sections describe the configuration of a Zorp firewall cluster. Although this is the most common scenario, other components of the Zorp Application Level Gateway System (for example, ZCV, ZAS) can also be clustered.

Warning
When creating a Zorp cluster, the ZMS managing the cluster must be on a dedicated machine, or on a Zorp host that is not part of the cluster. ZMS *cannot* be clustered.

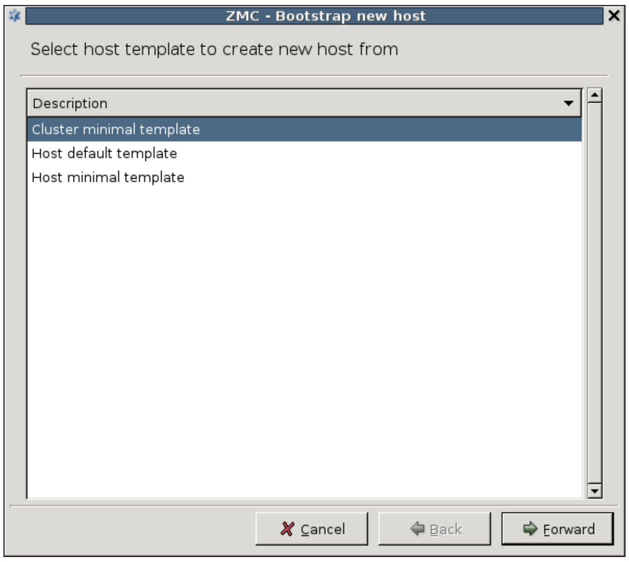
12.4.1. Procedure – Creating a new cluster (bootstrapping a cluster)

To create and bootstrap a new cluster, complete the following steps:

Note
As an alternative to creating a cluster and bootstrapping its first node, an existing Zorp host can also be converted into a cluster. For details, see *Procedure 12.4.4, Converting a host to a cluster (p. 299)*.

Step 1. Select the site that will include the new cluster from the configuration tree, and click on **New Host Wizard** in the **Management** menu.

Step 2. Select the **Cluster minimal template** and click on **Forward** to start bootstrapping the first node of the cluster. (Bootstrapping cluster nodes is very similar to bootstrapping individual Zorp hosts. For more information see *Chapter 4, Registering new hosts (p. 53)*.)

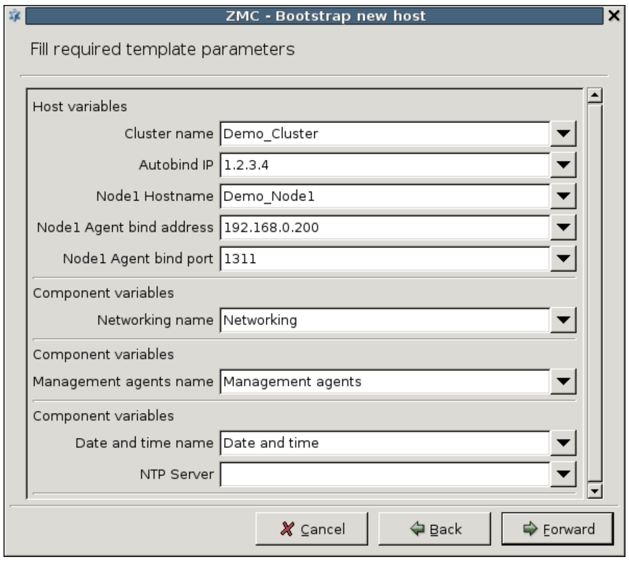

ZMC - Bootstrap new host

Select host template to create new host from

Description
Cluster minimal template
Host default template
Host minimal template

Figure 12.4. Bootstrapping a cluster

Step 3. Provide a name for the cluster (for example, `Demo_cluster`), as well as an **Agent Bind IP address**, an **Agent Bind IP port** and a **Hostname** (for example, `Demo_cluster_node1`) for the first node of the cluster. The node will accept connections from the ZMS agents on the specified **Agent Bind IP/Port** pair.


ZMC - Bootstrap new host

Fill required template parameters

Host variables

Cluster name	Demo_Cluster
Autobind IP	1.2.3.4
Node1 Hostname	Demo_Node1
Node1 Agent bind address	192.168.0.200
Node1 Agent bind port	1311

Component variables

Networking name	Networking

Component variables

Management agents name	Management agents

Component variables

Date and time name	Date and time
NTP Server	

Figure 12.5. Entering basic parameters

Step 4. The rest of the bootstrapping process is identical to bootstrapping a normal Zorp host, that is, create a certificate for the cluster, supply a one-time-password, and so on. For more information see *Chapter 4, Registering new hosts (p. 53)*.

Step 5. After bootstrapping the first node of the cluster, complete the following procedures as needed:

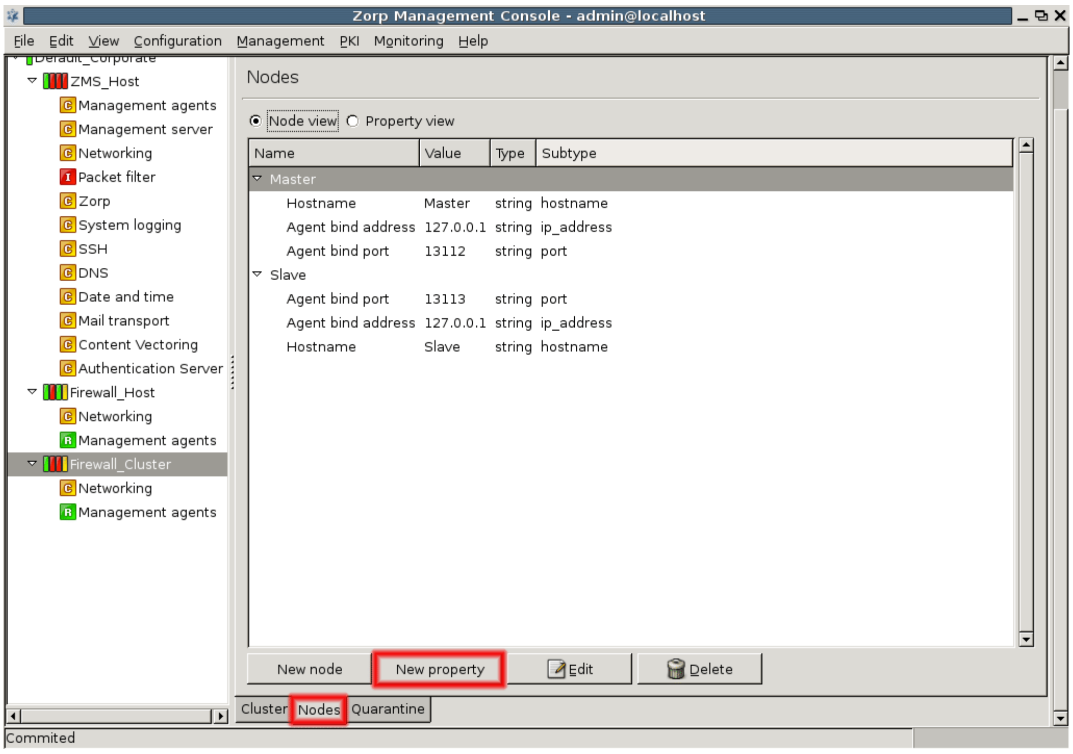
- Add additional nodes as required. For details, see *Procedure 12.4.3, Adding a new node to a Zorp cluster (p. 297)*.

- Adding new properties to the cluster. For details, see *Procedure 12.4.2, Adding new properties to clusters (p. 296)*.

12.4.2. Procedure – Adding new properties to clusters

As properties have to be used for all parameters that are different on each node, it is recommended to create all properties before adding the additional hosts to the cluster. Naturally, this is not required; properties can be defined any time. To add a new property to the cluster, complete the procedure below.

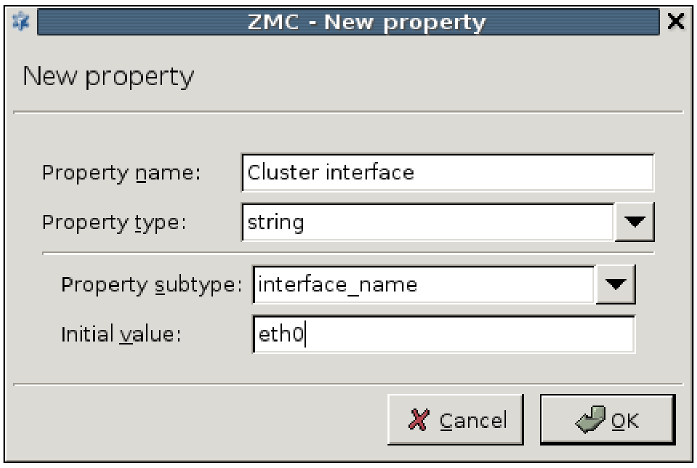
Step 1. Click the **New property** button on the **Nodes** tab of the cluster to define a new property.



Name	Value	Type	Subtype
Master			
Hostname	Master	string	hostname
Agent bind address	127.0.0.1	string	ip_address
Agent bind port	13112	string	port
Slave			
Agent bind port	13113	string	port
Agent bind address	127.0.0.1	string	ip_address
Hostname	Slave	string	hostname

Figure 12.6. Adding a new property

Step 2. Enter a name for the new property, and select the type and subtype of the property.



Property name: Cluster interface
Property type: string
Property subtype: interface_name
Initial value: eth0

Figure 12.7. Defining a new property

The possible property subtypes are the following.

- ip_address
- port
- ip_netmask
- interface_name
- hostname

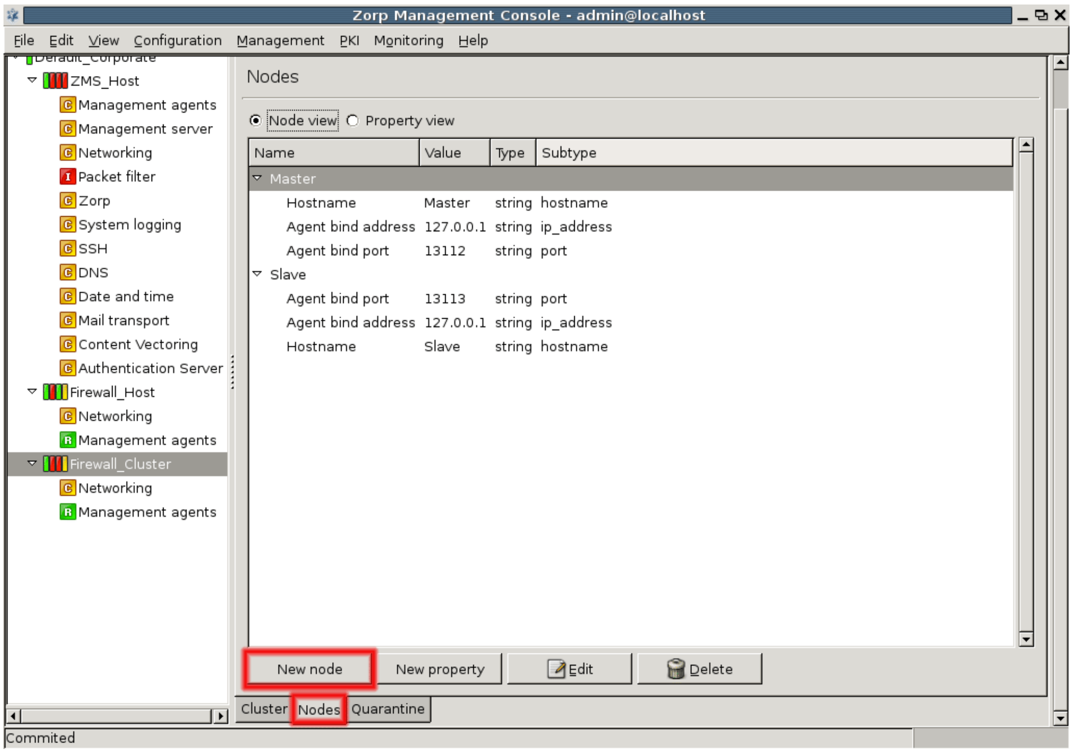
You can set initial values for the properties as well.

The new property is added to all nodes automatically. (Properties can be manipulated both in **Nodes** and **Properties** view.)

Step 3. Set the value of the new property for all nodes separately by clicking the **Edit** button.

12.4.3. Procedure – Adding a new node to a Zorp cluster

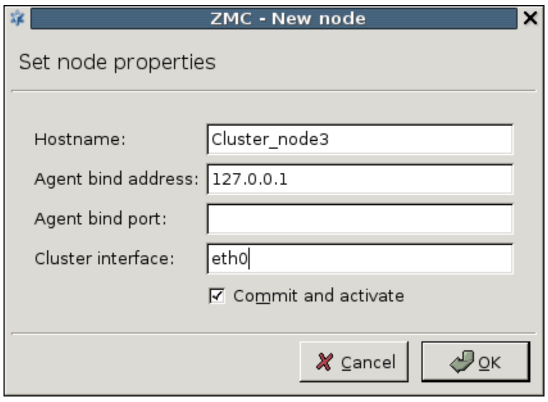
Step 1. Click the **New node** button on the **Nodes** tab of the cluster.



Name	Value	Type	Subtype
Master			
Hostname	Master	string	hostname
Agent bind address	127.0.0.1	string	ip_address
Agent bind port	13112	string	port
Slave			
Agent bind port	13113	string	port
Agent bind address	127.0.0.1	string	ip_address
Hostname	Slave	string	hostname

Figure 12.8. Adding a new node to a cluster

Step 2. Set the properties of the new node in the appearing window. Enter hostname, agent bind address and bind port, and any other properties that have been added to the cluster.



Hostname:	Cluster_node3
Agent bind address:	127.0.0.1
Agent bind port:	
Cluster interface:	eth0

☑ Commit and activate

Figure 12.9. Configuring the properties of the new node

Tip
It is recommended to use the default port setting.

Do not forget to check the **Commit and activate** checkbox to automatically commit the changes and connect to the new node. If this checkbox is not selected when the new node is created, the configuration must be committed into the ZMS database manually. Also, the node cannot be connected automatically, only through a recovery connection (see *Procedure 13.3.4, Configuring recovery connections (p. 348)*).

If you create a failover cluster, usually the second node is configured to be the slave node.

Step 3. Enter the one-time password and click the **OK** button to build up the connection.

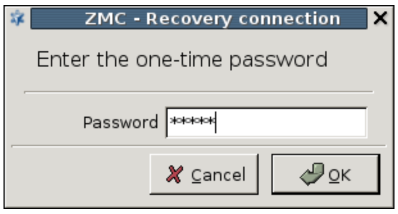


Figure 12.10. Entering the one-time-password

Details on the background procedures are provided in standalone text. Save the output with the **Save** button so that it can be analyzed later if needed. The text window shall look similar to this.

Note
You can check the status and connections of cluster nodes by selecting the Connections item in the **Management** menu.

After bootstrapping a cluster and adding new properties you can freely add the necessary components and configure the nodes according to your needs. Basically, the configuration procedure is similar to a Zorp host configuration. When configuring clusters using the **Heartbeat** component, proceed to *Section 12.5.3, Configuring Heartbeat (p. 301)*.

12.4.4. Procedure – Converting a host to a cluster

Existing Zorp hosts can also be converted into a cluster relatively easily. In this case the Zorp host will be converted into a node of the new cluster.

Warning
When a host is converted into a cluster, it retains all parameters that were set explicitely on the host. These parameters have to be replaced with links manually if needed. Typically, properties have to be created and links used for the `hostname`, `IP address` , and the `interface` parameters.

Step 1. Select the host you want to convert to a cluster.

Step 2. Select the Convert Host to Cluster in the **Management** menu.

Step 3. Enter a name for the cluster.

12.5. Heartbeat

12.5.1. Functionality of Heartbeat

In several cluster solutions (for example, in failover and multicast load balancing clusters) the nodes in the cluster must monitor each other to detect if one of the nodes fails.

Heartbeat is a tool monitoring the status of the nodes in a cluster. The **Heartbeat** components of the nodes send keep alive messages to the other node(s). When the node stops sending heartbeat packets it is assumed to be dead, and any services (resources) it was providing are taken over by the other node(s). For this functionality, you need to define the master and slave nodes, set encryption for the communication between the nodes, and also establish and configure a dedicated interface for the communication.

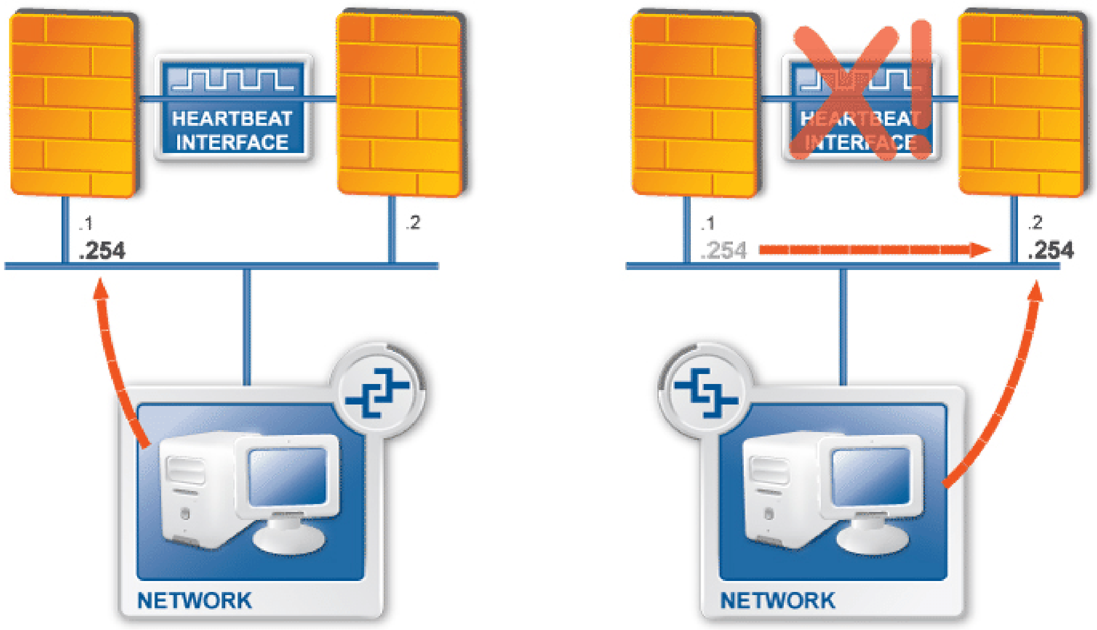


Figure 12.11. How Heartbeat works

Note

In order to use Heartbeat, the `heartbeat` package must be installed on all nodes of the cluster. Currently the Heartbeat package has to be installed manually by issuing the `apt install heartbeat-2` command as root from a command line.

It is recommended to encrypt the Heartbeat signals even if you are using a dedicated interface.

Heartbeat packets can be transferred through a serial null modem cable and / or Ethernet network, for example in case of geographically separated cluster nodes. If Ethernet is used, the heartbeat signals are UDP packets targeting a broadcast address.

Heartbeat instances are installed on all nodes. Both the master (active) and the slave (passive) nodes send heartbeat packets as a kind of keep-alive message across a dedicated interface. These messages enable the monitoring and, if needed, the takeover of each others' resources. When heartbeat packets are no longer received, the node is assumed to be dead, and any services (resources) it was providing are failed over to the other node.

Note

Unfortunately it is possible that the active node fails only partially, that is, although it stops sending heartbeat messages, it still replies to ARP request and retains the Service IP. Such situation results in two hosts — seemingly both functioning — owning the same IP on the LAN. To avoid such situation the death of the node has to be ensured by the integration of a STONITH (Shoot the Other Node In The Head) device, which practically turns off the power on the master node if it is dead.

It is also possible to install a hardware watchdog into the nodes of a cluster. A hardware watchdog is a small device that periodically receives some kind of signal (for example the heartbeat messages) from the computer — either from the kernel, or from a specific application. If the computer stops sending these signals, it is assumed to be dead, and the watchdog reboots the node.

Tip
Create a dedicated network for heartbeat messages using two Network Interface Cards (NIC) and a crossover Ethernet cable connecting them.

12.5.2. Heartbeat resources

Another important task of **Heartbeat** is that it manages the *Resources* of the cluster. A *resource* can be anything that is available on a node of the cluster and is relevant to the service it provides: it can be an IP address (for example, the Service IP of a fail-over cluster), a software running on the nodes (for example, Zorp), and so on. Actually these resources are what is taken over when a node fails. Heartbeat manages the resources through *resource scripts* — regular shell script files that can start and stop the resource on the node (for example, bring an interface into up/down state), and can return status information about the resource (for example, indicate whether the interface is in up or down state). How to configure heartbeat resources is described in *Procedure 12.5.4, Configuring Heartbeat resources (p. 308)*.

12.5.3. Configuring Heartbeat

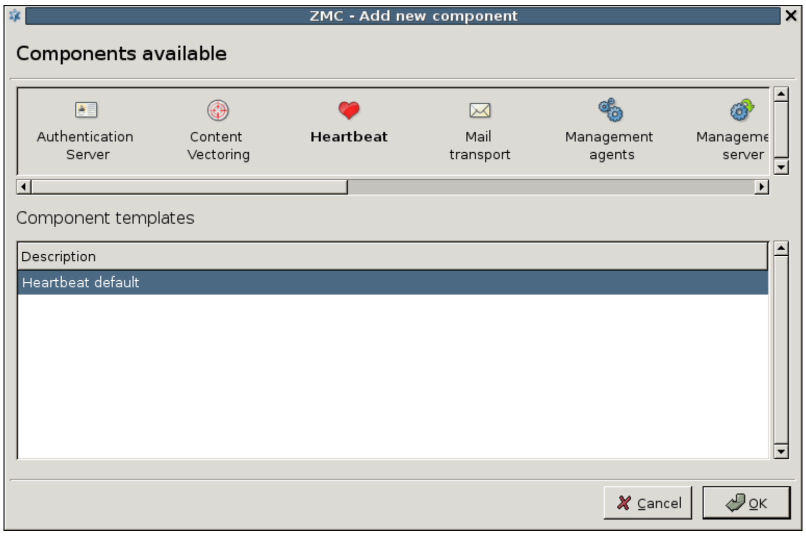
To configure the `Heartbeat` component, add it to the cluster first.

Note
The **Heartbeat** component can monitor multiple nodes, however, the **Heartbeat** ZMC component currently (version 3.4) supports only two nodes. It is possible to create and manage clusters containing three or more nodes in Zorp, but the **Heartbeat** configuration file has to be edited and managed manually in this case.

12.5.3.1. Procedure – Configure Heartbeat

Step 1. Select the cluster in the configuration tree and click the **New** button below the **Components in use** subwindow on the **Cluster** tab to add the `Heartbeat` component.

Step 2. Choose the **Heartbeat default** template and then change the component name, if needed.


ZMC - Add new component

Components available

Authentication Server | Content Vectoring | **Heartbeat** | Mail transport | Management agents | Manageme server

Component templates

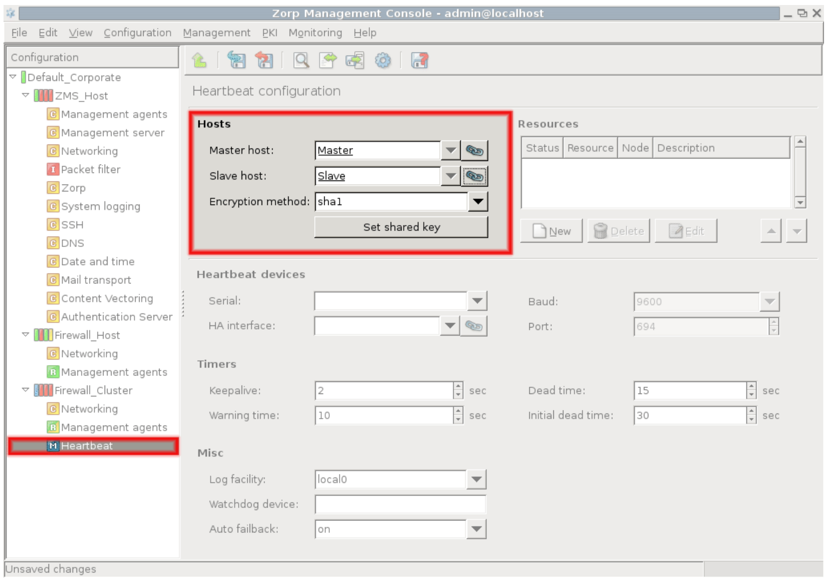
Description
Heartbeat default

Cancel | OK

Figure 12.12. Adding the Heartbeat component to the cluster

The `Heartbeat` component appears in the configuration tree.

Step 3. Define the master and slave hosts in the **Hosts** field of the Heartbeat main window. Each node must have a unique hostname.


Zorp Management Console - admin@localhost

Configuration
- Default_Corporate
  - ZMS_Host
    - Management agents
    - Management server
    - Networking
    - Packet filter
    - Zorp
    - System logging
    - SSH
    - DNS
    - Date and time
    - Mail transport
    - Content Vectoring
    - Authentication Server
  - Firewall_Host
    - Networking
    - Management agents
  - Firewall_Cluster
    - Networking
    - Management agents
    - Heartbeat

Heartbeat configuration

Hosts

Master host:	Master
Slave host:	Slave
Encryption method:	sha1

Set shared key

Resources

Status	Resource	Node	Description

New | Delete | Edit

Heartbeat devices

Serial:		Baud:	9600
HA interface:		Port:	694

Timers

Keepalive:	2 sec	Dead time:	15 sec
Warning time:	10 sec	Initial dead time:	30 sec

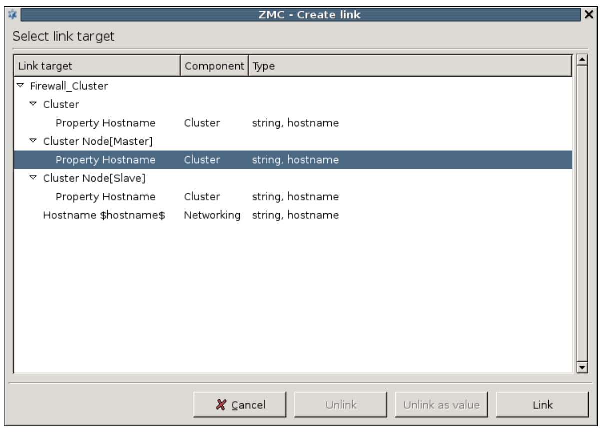
Misc

Log facility:	local0
Watchdog device:	
Auto failback:	on

Unsaved changes

Figure 12.13. Defining master and slave nodes

Note
It is recommended to use links when referring to nodes and their IP addresses. It is important to link to the properties of the node, and not to the properties of the cluster.


ZMC - Create link

Select link target

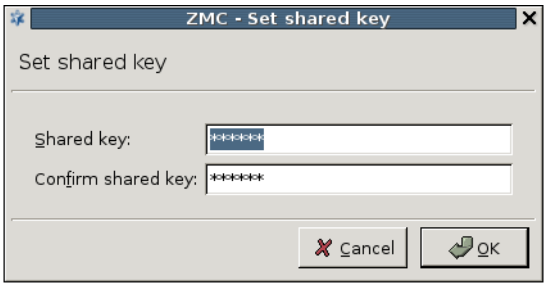
Link target	Component	Type
Firewall_Cluster		
Cluster		
Property Hostname	Cluster	string, hostname
Cluster Node[Master]		
Property Hostname	Cluster	string, hostname
Cluster Node[Slave]		
Property Hostname	Cluster	string, hostname
Hostname $hostname$	Networking	string, hostname

Figure 12.14. Using links for specifying the master and slave nodes

Step 4. Define encryption method for the Heartbeat messages.
The default encryption is SHA1, but MD5 and CRC are also possible.

Tip
It is recommended to use the default SHA1 encryption.

Step 5. Click the **Set shared key** button and give a password.


ZMC - Set shared key

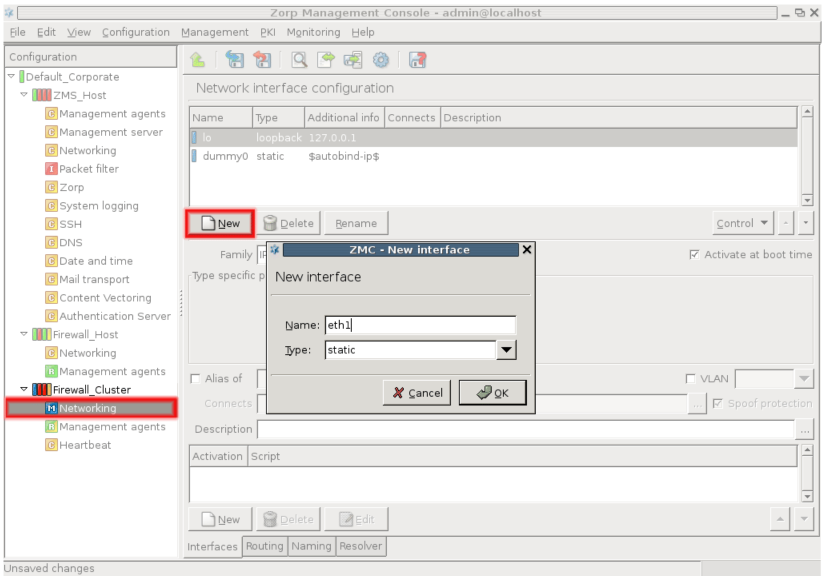
Set shared key

Shared key: ******
Confirm shared key: ******

Figure 12.15. Setting the shared key

Step 6. Now the communication channel used to transfer heartbeat messages between the nodes has to be configured. The following steps describe how to configure an Ethernet interface for this purpose. This involves using resources as well; the use of resources is described in *Procedure 12.5.4, Configuring Heartbeat resources (p. 308)* in detail.

Note
When using a serial interface to transfer the heartbeat messages between the two machines, a serial null-modem cable has to be used. It is recommended to use the highest possible communication speed (baud rate). For further information see the documentation of Heartbeat at *http://www.linux-ha.org/ConfiguringHeartbeat*.



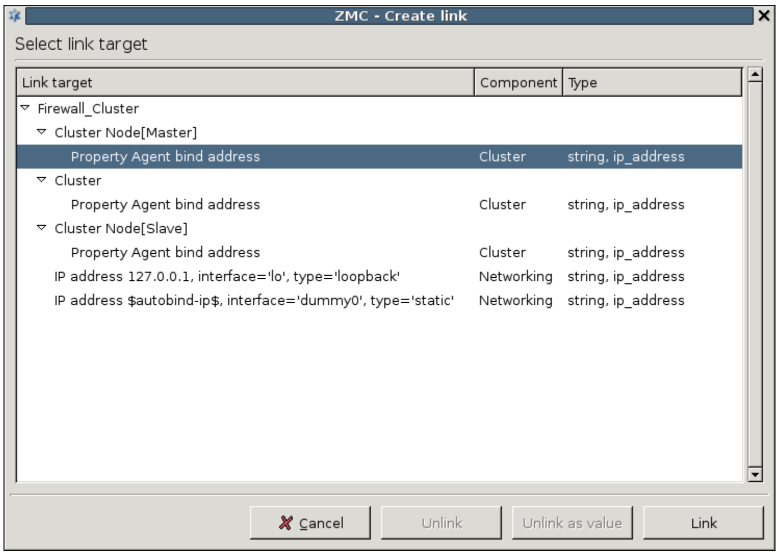
Name	Type	Additional info	Connects	Description
lo	loopback	127.0.0.1		
dummy0	static	$autobind-ip$		

Name: eth1
Type: static

Figure 12.16. Defining an Ethernet interface for Heartbeat

Go to the `Networking` component of the cluster, **Interfaces** tab. Create a new interface, for example, *`eth1`* for HA purposes using the **New** button.

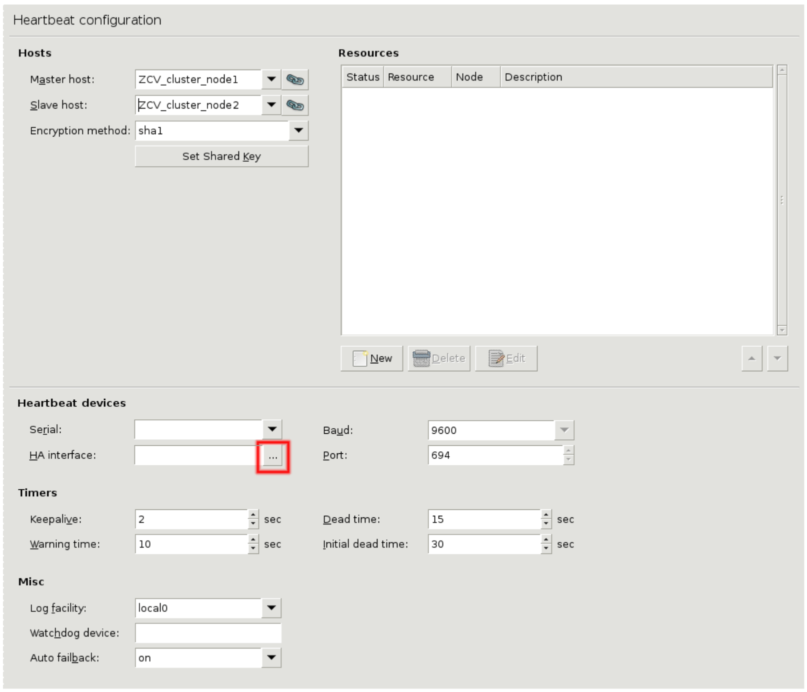
Step 7. Link the cluster HA IP address in the **Type specific parameters** field using the link icon. Enter the broadcast address as well.
Set **Netmask** and the **Network** fields, if no defaults are provided.



Link target	Component	Type
Firewall_Cluster		
Cluster Node[Master]		
Property Agent bind address	Cluster	string, ip_address
Cluster		
Property Agent bind address	Cluster	string, ip_address
Cluster Node[Slave]		
Property Agent bind address	Cluster	string, ip_address
IP address 127.0.0.1, interface='lo', type='loopback'	Networking	string, ip_address
IP address $autobind-ip$, interface='dummy0', type='static'	Networking	string, ip_address

Figure 12.17. Linking the bind address of the cluster to the Heartbeat interface

Step 8. Go to the `Heartbeat` component of the cluster and click **...** at the **HA interface** field. Click **New** and select the newly created interface as the HA interface for Heartbeat using the link icon or the drop down list.


Heartbeat configuration

Hosts

Master host: ZCV_cluster_node1
Slave host: ZCV_cluster_node2
Encryption method: sha1
Set Shared Key

Resources

Status	Resource	Node	Description

New Delete Edit

Heartbeat devices

Serial:
HA interface: ...
Baud: 9600
Port: 694

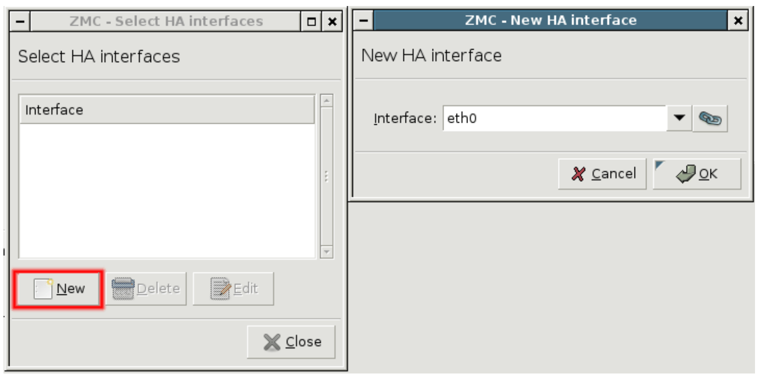
Timers

Keepalive: 2 sec
Warning time: 10 sec
Dead time: 15 sec
Initial dead time: 30 sec

Misc

Log facility: local0
Watchdog device:
Auto failback: on

Figure 12.18. Selecting the Heartbeat interface


ZMC - Select HA interfaces

Select HA interfaces

Interface

New Delete Edit

Close

ZMC - New HA interface

New HA interface

Interface: eth0

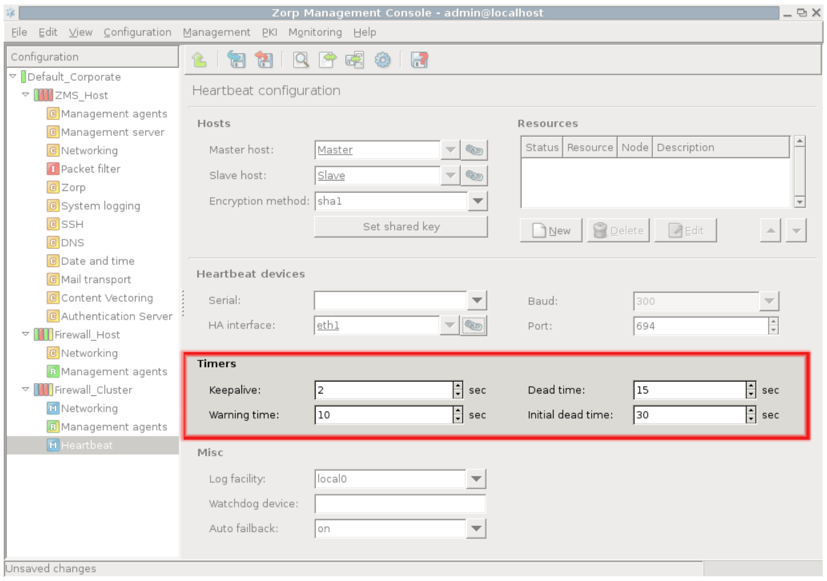
Cancel OK

Figure 12.19. Creating and naming a new Heartbeat interface

Note
Heartbeat can use two or more interfaces to avoid false takeovers when the HA interface of a node breaks down. If multiple HA interfaces are configured, the slave node becomes active only if all HA interfaces stop transmitting heartbeat messages.

12.5.3.2. Procedure – Configure additional Heartbeat parameters

Step 1. Set timers for Heartbeat in the **Timers** section.



Heartbeat configuration

Hosts

Master host:	Master
Slave host:	Slave
Encryption method:	sha1

Resources

Status	Resource	Node	Description

Heartbeat devices

Serial:		Baud:	300
HA interface:	eth1	Port:	694

Timers

Keepalive:	2 sec	Dead time:	15 sec
Warning time:	10 sec	Initial dead time:	30 sec

Misc

Log facility:	local0
Watchdog device:	
Auto failback:	on

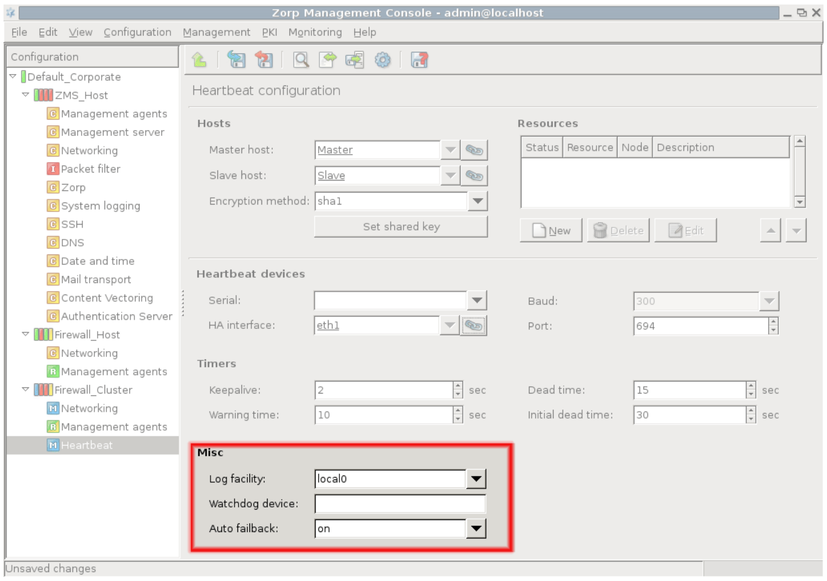
Figure 12.20. Configuring timing parameters

The following timers can be defined. Give all the values in seconds.

keepalive	The interval in seconds between subsequent Heartbeat signals.
warning time	The interval in seconds after which the slave node gives warning of the dead master node.
dead time	The interval in seconds after which the node is assumed to be dead.
initial dead time	First deadtime after system reboot in seconds.

Tip
It may take a while in some machines to have a fully functional network after a reboot. It is recommended to set the dead time twice as long as the warning time, and the initial deadtime twice as long as the dead time. For example, if the warning time is 20 seconds, the optimal dead time is 40 seconds and the optimal initial dead time is 80 seconds.

Step 2. Set other Heartbeat options in the **Misc** subwindow.



Heartbeat configuration

Hosts

Master host:	Master
Slave host:	Slave
Encryption method:	sha1

Resources

Status	Resource	Node	Description

Heartbeat devices

Serial:		Baud:	300
HA interface:	eth1	Port:	694

Timers

Keepalive:	2 sec	Dead time:	15 sec
Warning time:	10 sec	Initial dead time:	30 sec

Misc

Log facility:	local0
Watchdog device:	
Auto failback:	on

Figure 12.21. Configuring other Heartbeat parameters

Select the log target, watchdog device and decide about service reacquiry in case of master node recovery.

log facility	This option determines the target device for logging.
watchdog	This option determines which watchdog device is used, if any. For example, `/dev/watchdog`. Watchdogs can monitor the heartbeat messages of a node and reboot the node if it fails.
nice fallback	This option prevents the master node from reacquiring cluster resources after a failover in case the master node gets functional again. Enabling ***nice_failback*** can cause problems in certain situations. Suppose there is a two-node cluster with a master (*Node_A*) and a slave (*Node_B*) node. If *Node_A* fails, *Node_B* acquires its resources and uses a STONITH device to power off the *Node_A*. When *Node_A* recovers, the resources remain on *Node_B* if ***nice_failback*** is enabled. However, if now *Node_B* seems to fail, *Node_A* can power off *Node_B* only if two STONITH devices are intalled to the system (one to power off *Node_A*, one to power off *Node_B*).

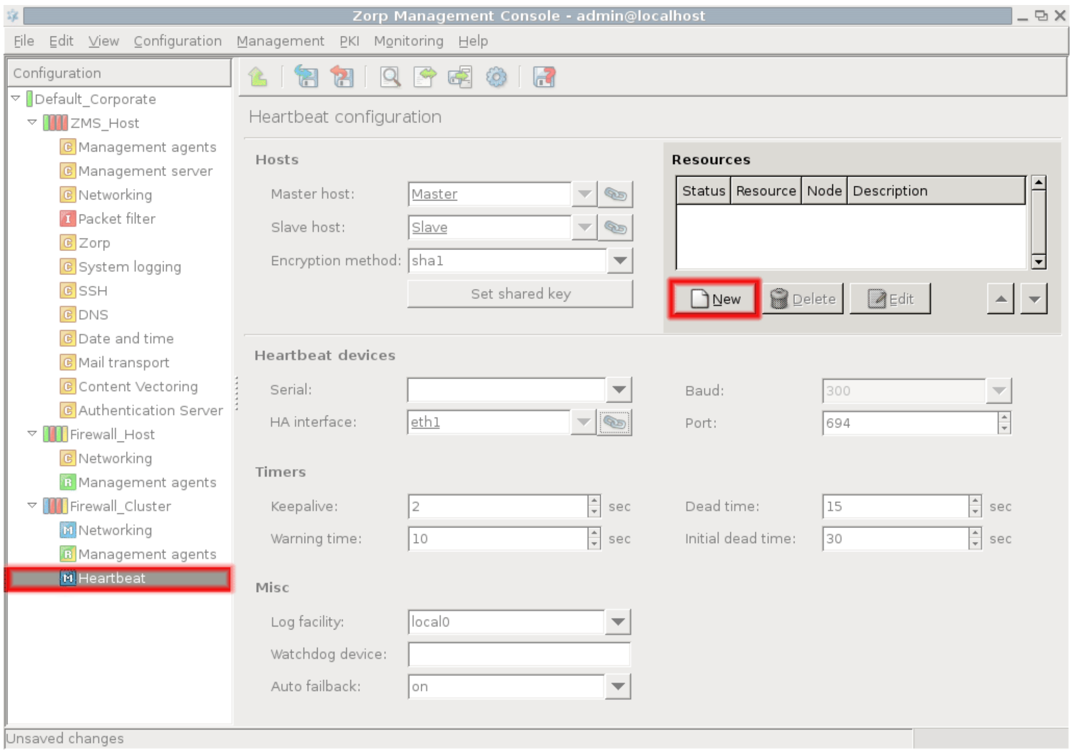
Tip
It is recommended to enable *nice_failback*, but after a takeover restore the original master-slave node hierarchy at the earliest possible time (for example, during the next maintenance break).

12.5.4. Procedure – Configuring Heartbeat resources

Configuring Heartbeat resources has two main steps: creating the resource scripts and adding these resources to the **Heartbeat** component in ZMC. The general procedure is as follows.

Step 1. Create a resource script, or modify an existing one. Heartbeat resource scripts have to be placed into the `/etc/ha.d/resource.d/` directory — this directory contains a number of resource script samples that cover the most commonly used scripts. Edit the parameters of the selected script.
The resource scripts installed by default are described in Step 3.

Step 2. Navigate to the **Heartbeat** component of the cluster. Resources can be managed in the **Resources** section of the panel. Click on **New** to add a new resource.

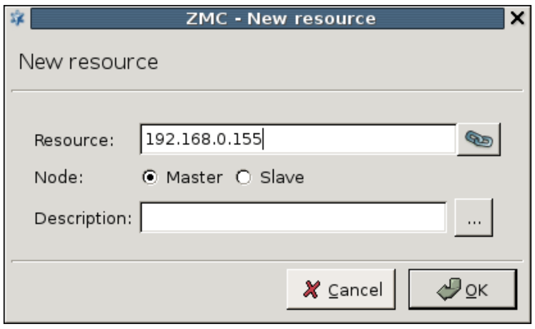


Master host:	Master
Slave host:	Slave
Encryption method:	sha1
Serial:	
HA interface:	eth1
Baud:	300
Port:	694
Keepalive:	2 sec
Dead time:	15 sec
Warning time:	10 sec
Initial dead time:	30 sec
Log facility:	local0
Watchdog device:	
Auto failback:	on

Status	Resource	Node	Description

Figure 12.22. Adding new Heartbeat resources

Step 3. Enter a name for the resource — this has to be the name of the resource script.



Resource: 192.168.0.155
Node: [✓ Master] Slave
Description:

Figure 12.23. Configuring Heartbeat resources

The following resource scripts are available by default (see the actual scripts in the `/etc/ha.d/resource.d` directory of the nodes for details):

- **IPaddr**: This option adds/removes an alias IP address (that is, the Service IP).

> **Note**
> If only an IP address is specified as the resource name, it is automatically considered as an *`IPaddr`* resource.

- **IPsrcaddr**: It manages the preferred source address associated with packets which originate on the localhost and are routed through the default route.
- **MailTo**: Send an e-mail to the system administrator whenever a takeover occurs.
- **SendArp**: It is an alternative to the **IPaddr** resource to send gratuitous ARP for an IP address on a given interface without adding the address to that interface. It shall be used if for some reason gratuitous ARP is required to be sent for addresses managed by **IPaddr** on an additional interface.

Step 4. The node on which the resource is active by default can be selected using the **Master/Slave** radio buttons.

12.5.5. Procedure – Configuring a Service IP address

The following example describes how to create an alias network interface and configure it as the Service IP of a cluster.

Step 1. Navigate to the **Networking** component of the cluster and click on **New** in the **Network interface configuration** section.

Step 2. Enter a name for the alias interface (for example, *`eth1:1`*). For further information on alias interfaces see *Section 5.1.2, Configuring virtual networks and alias interfaces (p. 68)*.

Step 3. Enter the *`IP address`*, `Netmask`, `Broadcast address`, and so on parameters into the respective fields. The *`IP address`* specified here will be the Service IP of the cluster, to be accessible by the clients.

> **Warning**
> Make sure the **Activate at boot time** checkbox is unchecked.

Step 4. Create a new resource by clicking the **New** button in the **Resources** section of the **Heartbeat** component. Enter the Service IP address. It is recommended to use linking to the IP address of the alias interface created in the previous steps. Also select the node where the resource is active using the **master** and **slave** radio buttons.

12.6. Keepalived for High Availability

12.6.1. Functionality of Keepalived

Beside the so far recommended high availability (HA) solution (**Heartbeat**), a new and more modern approach for Virtual IP address transition between cluster nodes is available in the Zorp Management Console (ZMC), namely, the **Keepalived** solution.

Keepalived offers a framework not only for enabling high availability but for load balancing as well. Its load balancing solution relies on the Linux Virtual Server (IPVS) kernel module, while its HA solution is based on Virtual Router Redundancy Protocol (VRRP). VRRP protocol provides automatic assignment of available IP routers to ensure resilient routing paths.

12.6.2. Prerequisites for configuring Keepalived

- The **Keepalived** component can only be added to cluster configuration.
- Install the **Keepalived** package in cluster hosts as follows:

  ```
  apt install keepalived
  ```

  It is available in Ubuntu Bionic main repository.

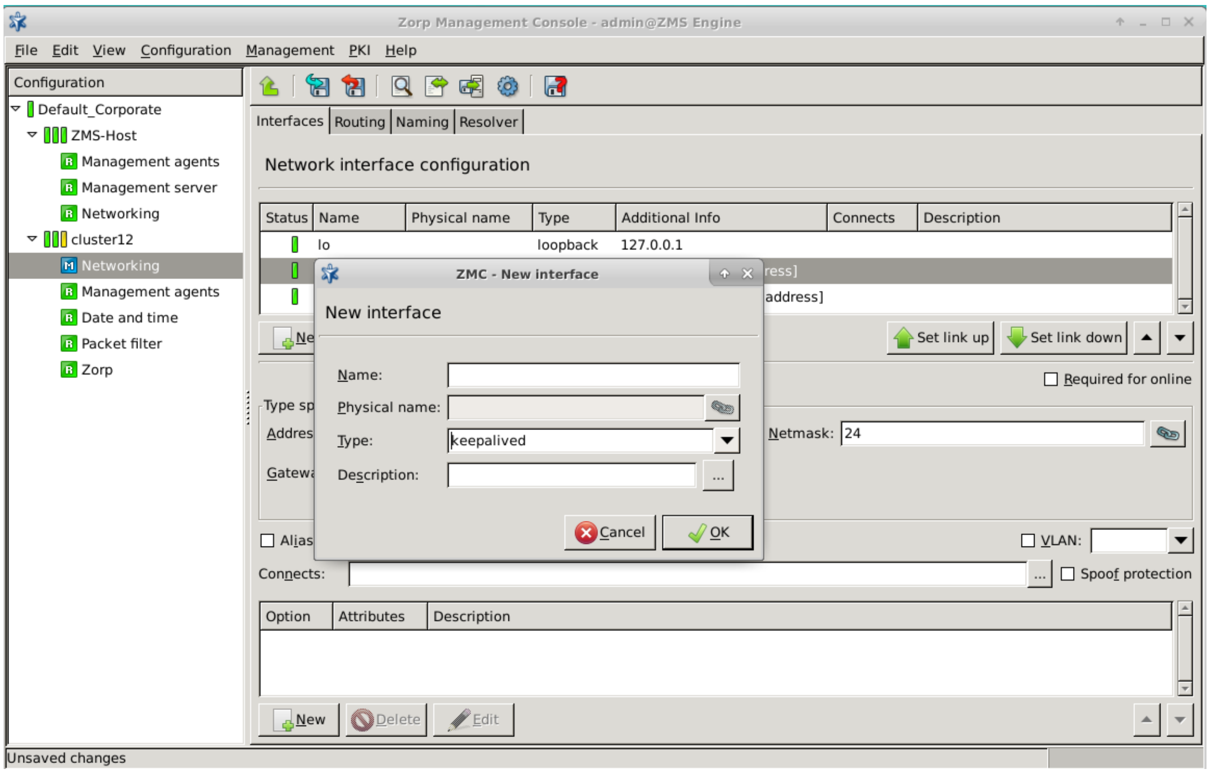
12.6.3. Configuring Keepalived

To configure a `Keepalived` HA cluster, the suggested steps are as follows:

12.6.3.1. Procedure – Configure Keepalived

Step 1. Create cluster configuration with configured network interfaces.

Step 2. Add Virtual IP addresses to the Networking component via setting the interface type to **keepalived**. Type-specific parameters are: 'Address' and 'Netmask'. They are the same as in case of 'static' interface type.



Status	Name	Physical name	Type	Additional Info	Connects	Description
🟩	lo		loopback	127.0.0.1		

Name:
Physical name:
Type: keepalived
Description:
Netmask: 24

Option	Attributes	Description

Figure 12.24. Setting interface type to 'Keepalived' in the Networking component

Note
If the type of the interface is **keepalived**, it must be an alias interface of an existing interface.

If the type of the interface is **keepalived** in the Networking component, it cannot be disabled. If the Virtual IP Address shall be disabled, it shall be set in the configuration of the **Keepalived** component.
When the Networking component is restarted, the Virtual IPs are dropped from the existing interfaces. To avoid this, add the *keep-configuration* interface option with the *static* value to those used physical interfaces which have *keepalived* alias interfaces on them.

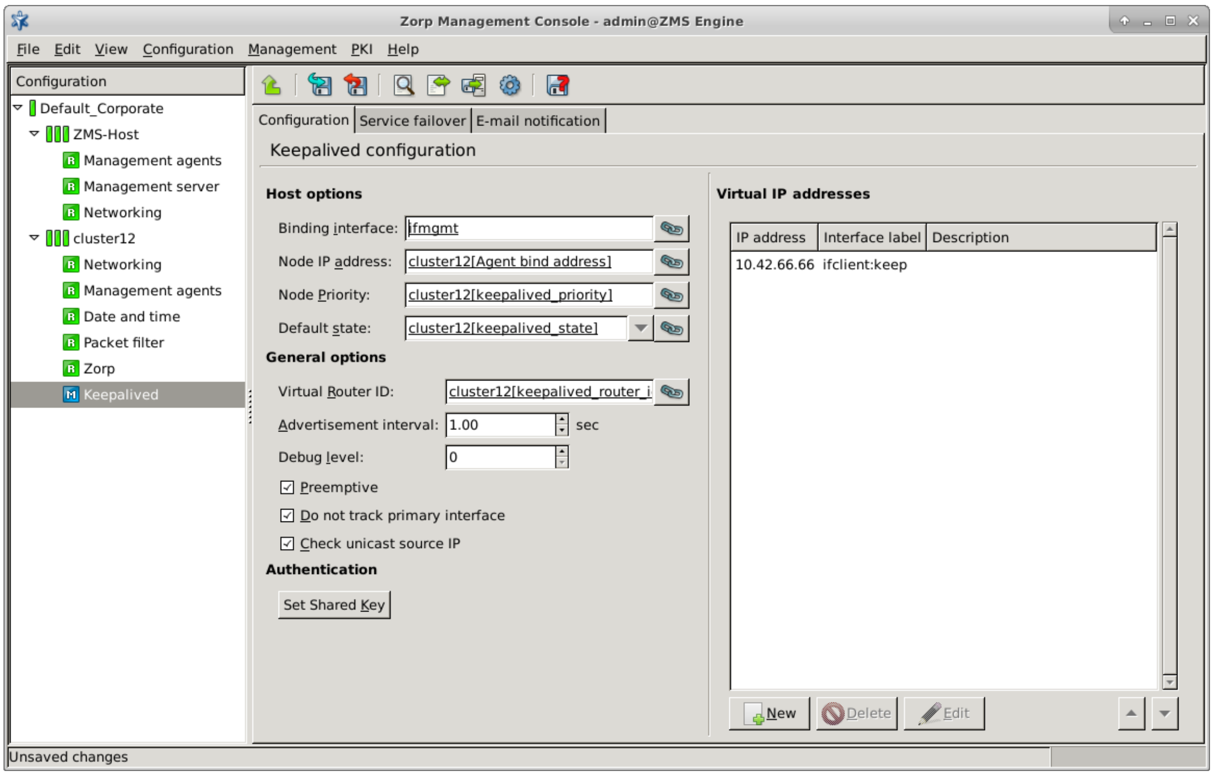
Also, if the *static* value is set, then after any change made on these interfaces, the old values will not be removed at the restart of the Networking component, but new values will be added (for example, IP, subnet). Temporarily turning the `keep-configuration` parameter to *no* and restarting the node is not advised, because the networking restart will remove all settings added by other sources too. It is recommended to reboot the node after these values have been changed, or to configure these changes manually, and skip restart. For details on configuring interface options, see *Procedure 5.1.6.3.1, Configuring interface parameters (p. 78)*.

Step 3. Add **keepalived** component to cluster configuration.

Step 4. Select the cluster in the configuration tree and click the **New** button below the **Components in use** subwindow on the **Cluster** tab to add the `Keepalived` component.

Step 5. Choose the **Keepalived default** template and change the component name, if needed.
The `Keepalived` component appears in the configuration tree.

Step 6. Set the configuration options for the **Keepalived** component under the **Configuration** tab.


Zorp Management Console - admin@ZMS Engine

File Edit View Configuration Management PKI Help

Configuration
- Default_Corporate
  - ZMS-Host
    - Management agents
    - Management server
    - Networking
  - cluster12
    - Networking
    - Management agents
    - Date and time
    - Packet filter
    - Zorp
    - Keepalived

Configuration | Service failover | E-mail notification

Keepalived configuration

Host options

Binding interface: ifmgmt

Node IP address: cluster12[Agent bind address]

Node Priority: cluster12[keepalived_priority]

Default state: cluster12[keepalived_state]

General options

Virtual Router ID: cluster12[keepalived_router_i

Advertisement interval: 1.00 sec

Debug level: 0

☑ Preemptive

☑ Do not track primary interface

☑ Check unicast source IP

Authentication

Set Shared Key

Virtual IP addresses

IP address	Interface label	Description
10.42.66.66	ifclient:keep	

New Delete Edit

Unsaved changes

Figure 12.25. The configuration options for Keepalived component under Configuration tab

The configuration options are as follows:

- Binding interface:
  The name of the interface, where Keepalived binds on.

- Node IP address:
  This option must be a linked cluster property of an IPv4 or IPv6 address, which is used as source address in VRRP packets and also for unicast peer IP purposes.

  Example:

  A firewall cluster with three nodes and with their Node IP address cluster properties' value:

  - node-1: 10.0.0.1
  - node-2: 10.0.0.2
  - node-3: 10.0.0.3

  In this case, for node-2, the unicast source IP option for Keepalived is as follows: 10.0.0.2. The unicast peer IP addresses are as follows: 10.0.0.1 and 10.0.0.3

- Node priority:
  It defines the VRRP priority of the cluster nodes.

Note
It can be set to the same value for all nodes, or linked via cluster property to be different on all nodes.

- Default state:
  The start-up default state of the nodes.

  Note
  It can be set to the same value for all nodes, or linked via cluster property to be different on all nodes.

  Note
  In case of non-preemptive configuration, the state for all nodes shall be BACKUP.

- Virtual Router ID:
  The value for the VRRP Virtual Router Identifier (VRID).

  Note
  It can be set to the same value for all nodes. If it is linked via cluster property, it can be used for grouping nodes.

- Debug level:
  This option sets the debug level of the Keepalived VRRP module between the values 0 and 4.

- Preemptive:
  This option enables or disables VRRP RFC preemption. If it is disabled, it allows the lower priority machine to maintain the master role, even when a higher priority machine comes back online.

- Do not track primary interface:
  It ignores VRRP interface faults. It is useful for cross-connect VRRP configurations.

- Check unicast source IP:
  It checks whether the source address of a unicast packet is a unicast peer.

- Set shared key:
  It is the authentication password used in VRRP packets.

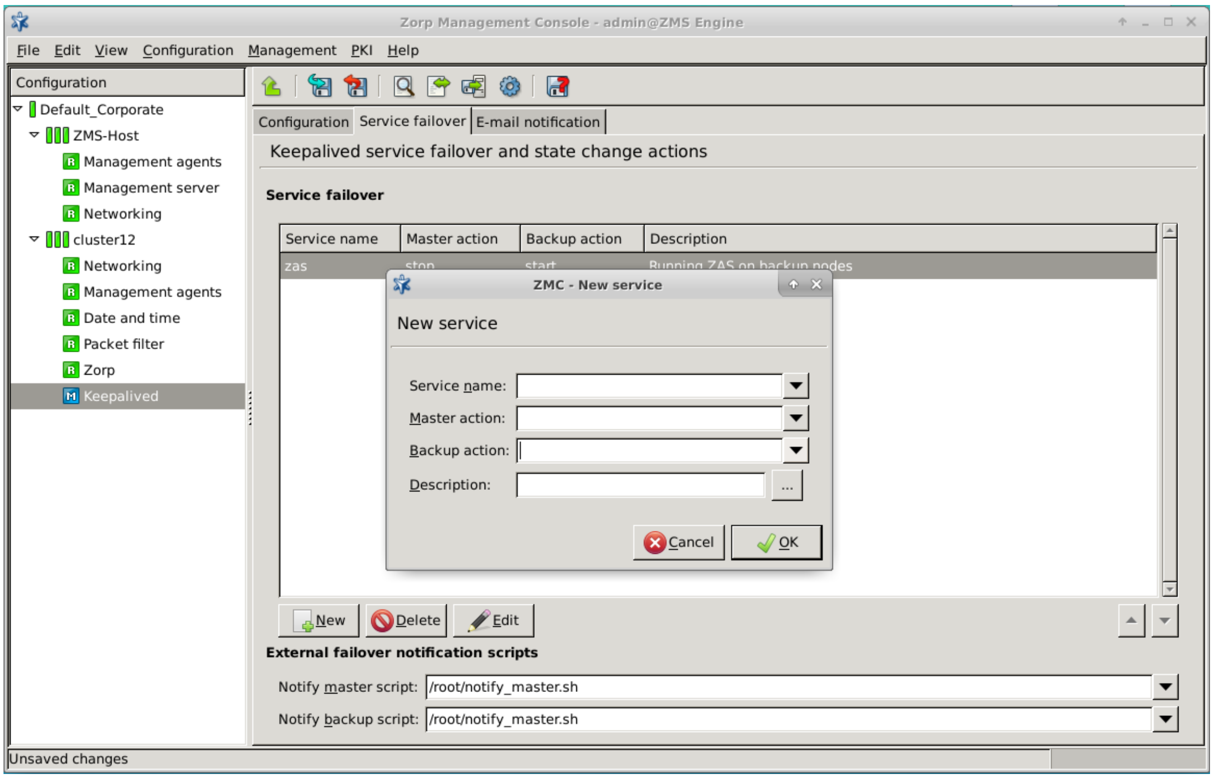
  Note
  Keepalived truncates passwords longer than 8 character.

- Virtual IP Addresses:
  This table contains the configured Virtual IP Addresses. The addresses shall be in the order of configuration precedence. The table can contain only linked IP addresses, which are configured in Networking, on interfaces, which have the **type** value Keepalived.

> **Note**
> Because of the limitations of the VRRP protocol, the first VRRP packet can contain the maximum of 20 IP addresses. The rest of the Virtual IP Addresses are defined later via extra packets.

> **Note**
> In case of mixing IPv4 and IPv6 Virtual IP Addresses there is a limitation of the VRRP protocol: In the first VRRP packet, the IP addresses must be in the same address family as that of the first item. The IP addresses with different address families are defined later via extra packets.

Step 7. Configure the following options in **Keepalived component**, under **Service failover** tab.



Keepalived service failover and state change actions

Service failover

Service name	Master action	Backup action	Description
zas	stop	start	Running ZAS on backup nodes

ZMC - New service

New service

Service name:	
Master action:	
Backup action:	
Description:	

External failover notification scripts

Notify master script:	/root/notify_master.sh
Notify backup script:	/root/notify_master.sh

Figure 12.26. Configuring Keepalived component under Service failover tab

- Service :
  In this table systemd service actions can be configured, which can be executed after the change of the state on the new master or backup nodes.

  Service actions are: start, stop, restart and reload.

Note
Consider disabling any of the listed systemd services to avoid the situation, when unrequested actions take place, in some cases even parallelly. For example, when the service is running on both nodes instead of running on only one node.

Note
Service names are suggested in the drop-down menu, according to the available modules, added in the cluster configuration. Free-text service names can also be added.

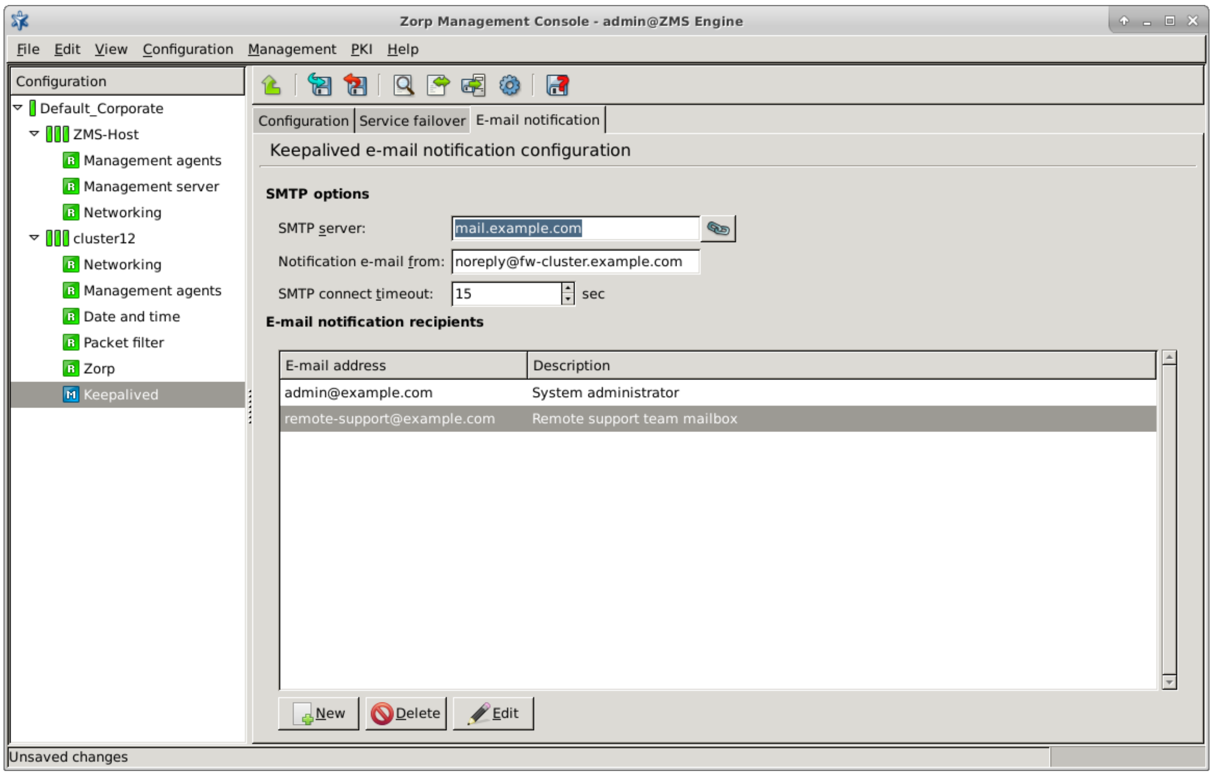
Note
It is not necessary to set action both for the master and the backup node. It can be set only for one of them.

- External failover notification scripts:
  User scripts can be given or selected from a list, which is executed on the hosts, when the change of the state has been executed.

Warning
There are strict requirements with regards to the rights of the selected script files. Without the necessary rights no action will be executed. The file must be owned by the root user and group, and there must not be write right for any other user. Neither shall be there writing right for any other level of the file path, except for the root user and group.

Note
In the listed drop-down menus a file can be selected, which is managed in a Freetext plugin for this Cluster.

Step 8. Configure the following items in **Keepalived component**'s **E-mail notification** tab.



Keepalived e-mail notification configuration

SMTP options

SMTP server:	mail.example.com
Notification e-mail from:	noreply@fw-cluster.example.com
SMTP connect timeout:	15 sec

E-mail notification recipients

E-mail address	Description
admin@example.com	System administrator
remote-support@example.com	Remote support team mailbox

Unsaved changes

Figure 12.27. Configuring Keepalived component under E-mail notification tab

- SMTP server:
  It is the IP address or domain name of SMTP server to use.

> **Note**
> Optional port parameter can be added, the default value is 25.

- Notification email from:
  It is the information in the header field on the address the e-mail is received from.
- SMTP connect timeout:
  It is the SMTP server connection timeout in seconds.
- E-mail notification recipients:
  It is the list of e-mail addresses, where the notifications on the change of state shall be sent.

> **Tip**
> Maintenance tip: if the firewall administrator wants to manually change the state of the cluster nodes ("switch over") it can be done as follows:
>
> - Restart (or in case of preemptive configuration, stop) the Keepalived service with Control Service on those nodes, which are not meant to be master nodes.
>
> Also consider Virtual Router IDs: if not all nodes have the same ID, restart (or stop) Keepalived service only on those nodes, which have the same ID.

Note
After completing Keepalived configuration, **Packet filter** component may be Invalidated, if **Keepalived packet filter rule** entry has been added/changed. The created rule allows VRRP traffic from all possible node IP addresses to enter the cluster hosts. The Virtual Router ID helps to identify the relevant VRRP packets in case of multiple node groups.

12.6.4. Configuration examples and best practices for Keepalived configuration

12.6.4.1. Procedure – Simple Cluster with 2 nodes

Step 1. Link the Node IP Address to the proper cluster property, which contains the IP addresses to be used for Keepalived VRRP traffic.

Step 2. Set **Node Priority** and **Virtual Router ID** to a fixed value, for example, to 100.

Step 3. Set the Default state to the value BACKUP and do not set Preemptive option (recommended steps).

Step 4. Enter an 8-character-long random string at Set Shared Key option.

Step 5. Link Virtual IP addresses, which have been configured in the Networking component.

The basic configuration is now done, it can be uploaded and Keepalived can be started.

12.6.4.2. Procedure – Testing or Pilot node

If there is an extra node in the cluster, for testing or piloting purpose, which is not planned to be used in the live keepalived configuration, then 2 solutions are suggested:

Step 1. Do not upload Keepalived configuration and do not start Keepalived service on that specific node. This trivial solution needs the active attention of the firewall administrator, when uploading Keepalived configuration or managing the service.

Step 2. Create and link a new cluster property with keepalived_router_id type.
Set the value of the new cluster property to the same for the actively used nodes and to a different value for the testing nodes.

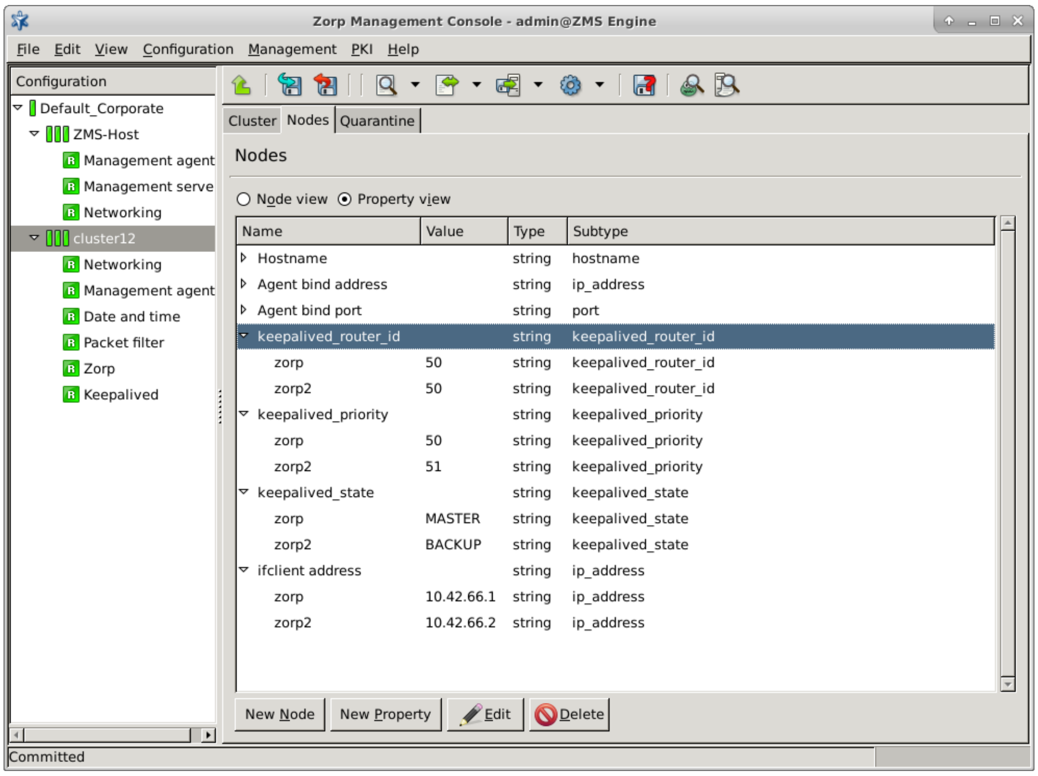
In this case, the testing nodes will be in a different VRRP group and they will not ever get MASTER state in the cluster.

12.6.4.3. Procedure – Multiple backup nodes

It is possible to have multiple backup nodes in the same VRRP group. If the nodes do not have the same hardware or the nodes differ in computing speed, it is suggested, to set the **Node priority** via linking **keepalived priority**-typed cluster property.

12.6.4.4. Procedure – Multiple VRRP groups in the same cluster

If there are at least 2N nodes in the cluster, where N > 2 and the nodes are logically paired.



Name	Value	Type	Subtype
Hostname		string	hostname
Agent bind address		string	ip_address
Agent bind port		string	port
keepalived_router_id		string	keepalived_router_id
zorp	50	string	keepalived_router_id
zorp2	50	string	keepalived_router_id
keepalived_priority		string	keepalived_priority
zorp	50	string	keepalived_priority
zorp2	51	string	keepalived_priority
keepalived_state		string	keepalived_state
zorp	MASTER	string	keepalived_state
zorp2	BACKUP	string	keepalived_state
ifclient address		string	ip_address
zorp	10.42.66.1	string	ip_address
zorp2	10.42.66.2	string	ip_address

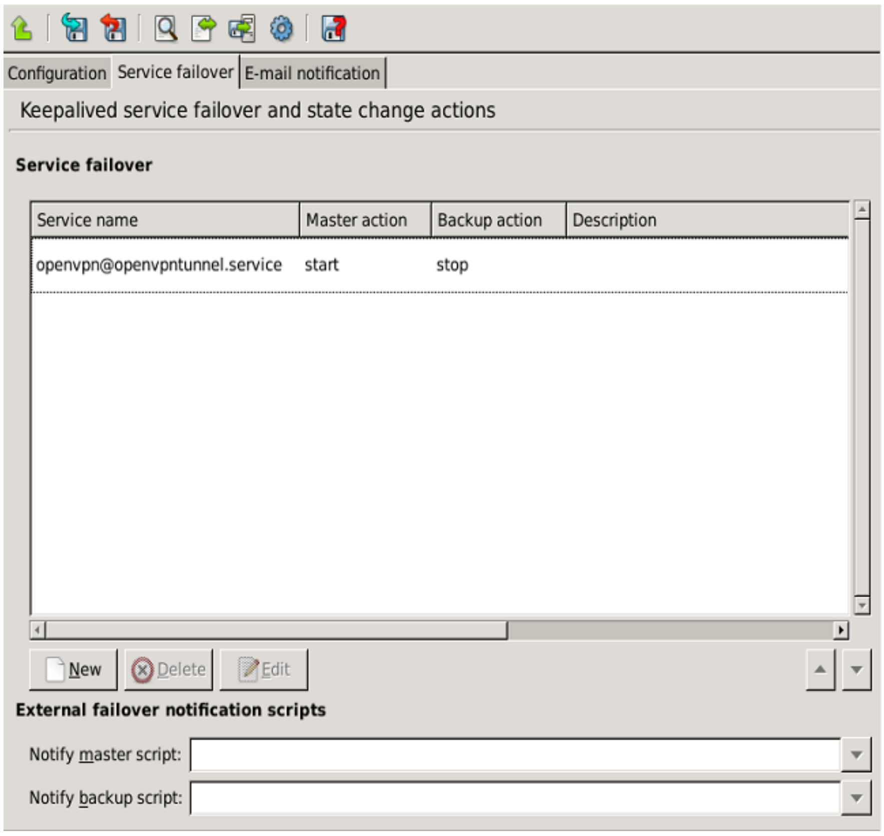
Figure 12.28. Multiple nodes in the same cluster

Step 1. Create a cluster property with keepalived_router_id type, and the value of the property shall be the same for the nodes, those are paired to each other.

Step 2. Also create a cluster property per Virtual IP Address, and set its value to the same IP address per host group. This cluster property shall be linked in the **Networking** component, when adding Keepalived type of interface.

12.6.4.5. Procedure – Managing individual OpenVPN tunnels

From Zorp version 7 on, it is possible to reference any individual OpenVPN tunnel handled by keepalived as an openvpn@tunnelname.service.


Configuration | Service failover | E-mail notification

Keepalived service failover and state change actions

Service failover

Service name	Master action	Backup action	Description
openvpn@openvpntunnel.service	start	stop	

External failover notification scripts

Notify master script:

Notify backup script:

Figure 12.29. OpenVPN tunnel service

12.7. Availability Checker

Having the availability status of some pre-selected target addresses can be advantageous during performing stateful failover HA functionality. Knowing beforehand that a certain address is not available and new connections shall not be attempted to them, can eliminate waiting time. This functionality is implemented in a service daemon named *failcheckd* which can monitor IP addresses with different methods and make this availability information visible for the firewall.

12.7.1. Prerequisites for configuring the Availability Checker plugin

Make sure that the *failcheckd* service is enabled:

```
systemctl enable failcheckd.service
```

12.7.2.1. Procedure – Configuring the Availability Checker

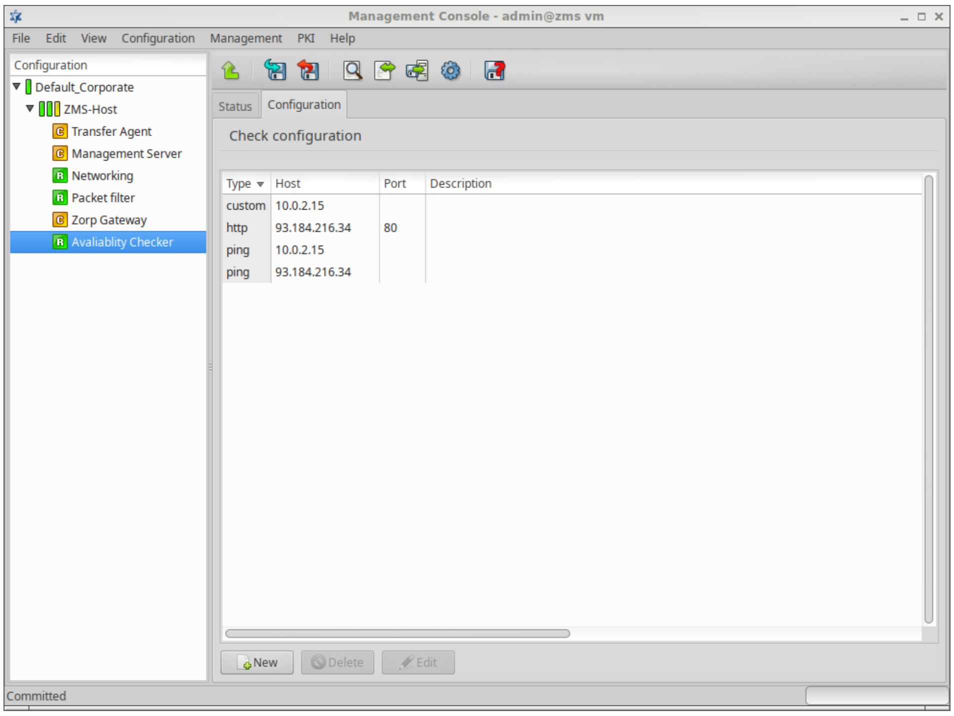
Complete the following steps in order to add *Availability Checker* component to the configuration.

Step 1. Select the host in the configuration tree and click the **New** button below the *Components in use* subwindow on the *Host* tab to add the *Availability Checker* component.

Step 2. Choose the **Default** template and change the component name, if needed.

Step 3. The *Availability Checker* component appears in the configuration tree.

Step 4. Set the configuration options for the *Availability Checker* component under the **Configuration** tab.



Check configuration

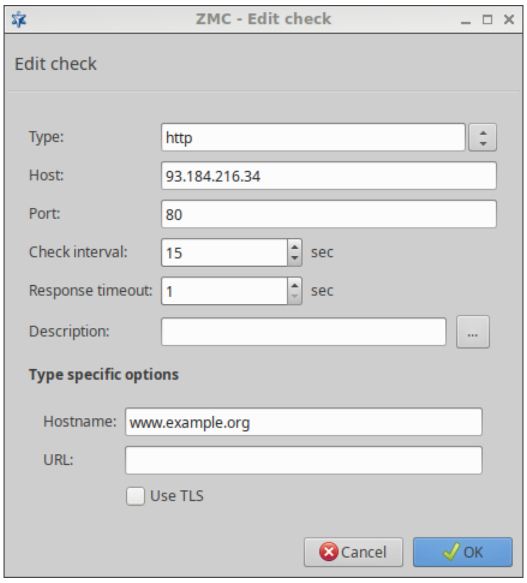
Type	Host	Port	Description
custom	10.0.2.15		
http	93.184.216.34	80	
ping	10.0.2.15		
ping	93.184.216.34		

Figure 12.30. Configuration options for Availability Checker

New, monitored target addresses can be added as new checks, specifying the followings:

- the check method (Type)
- host address
- port
- check interval
- response timeout
- other, method-dependent options

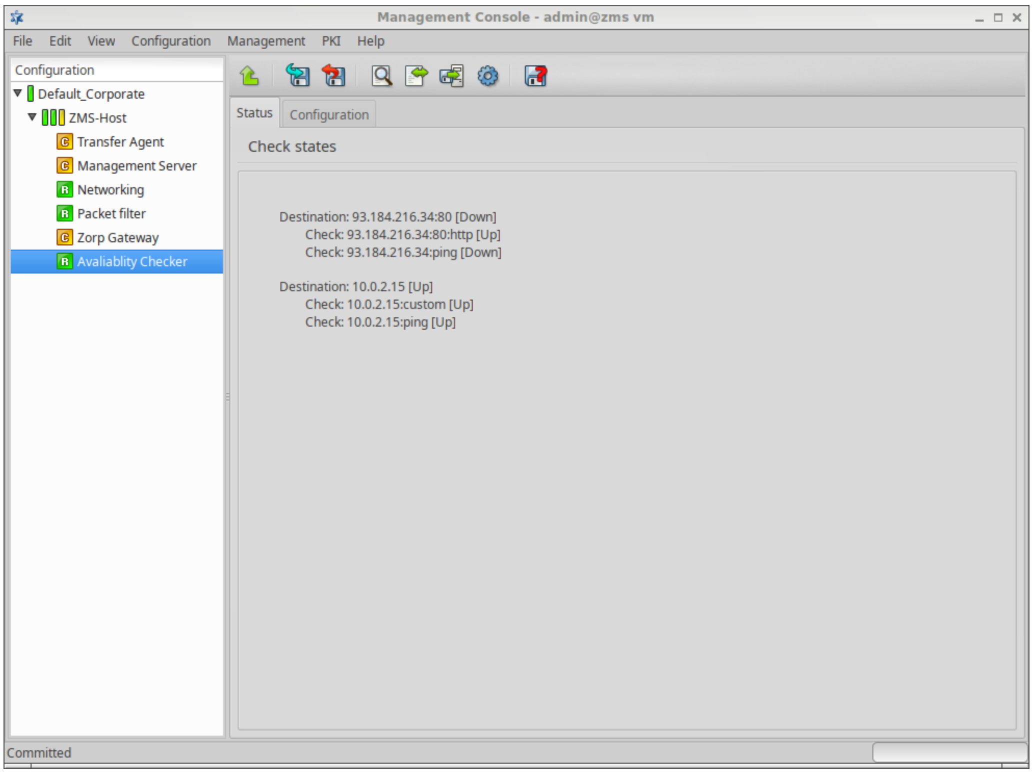
Apart from the *ping*, *TCP*, and *HTTP* methods, there is a *custom* type method which can use any executable program's return value. The returned values shall be set so, that the value zero shows success, the other values represent failure. Programs used in *custom* checks, get the *Response timeout* value as the value of the `--timeout RESPONSE_TIMEOUT` first command line parameter. Executable programs are terminated with the *SIGTERM* signal in 5 seconds after the timeout value has elapsed as set in the `Response timeout` parameter.



Type: http
Host: 93.184.216.34
Port: 80
Check interval: 15 sec
Response timeout: 1 sec
Description:
Type specific options
Hostname: www.example.org
URL:
Use TLS

Figure 12.31. Check details

Step 5. Check the **Status** tab. Following the upload of the plugin configuration and the restart of the service, the *Status* tab shows the actual status of the checks. The status of each target address comes from the aggregated status values of the checks of the certain address.



```
Destination: 93.184.216.34:80 [Down]
     Check: 93.184.216.34:80:http [Up]
     Check: 93.184.216.34:ping [Down]

Destination: 10.0.2.15 [Up]
     Check: 10.0.2.15:custom [Up]
     Check: 10.0.2.15:ping [Up]
```

Figure 12.32. Status of checks

Note

The auto refresh time of the *Status* tab can be globally configured in the **Program status** section of **Edit** menu in **Preferences...** option.

Chapter 13. Advanced ZMS and Agent configuration

The Zorp Management Server (ZMS) is the central component of the Zorp Management System. It governs all configuration settings, manages the firewall services through agents and handles database functions.

ZMS provides a tool for the complete control and maintenance of the Zorp firewalls entirely. It is possible to create new firewall configurations, and navigate them to the firewall nodes. ZMS stores these configurations in an associated XML database making them available for later administrative operations.

Communication with the Zorp firewall software is realized by the Transfer Agent responsible for accepting and executing configuration commands.

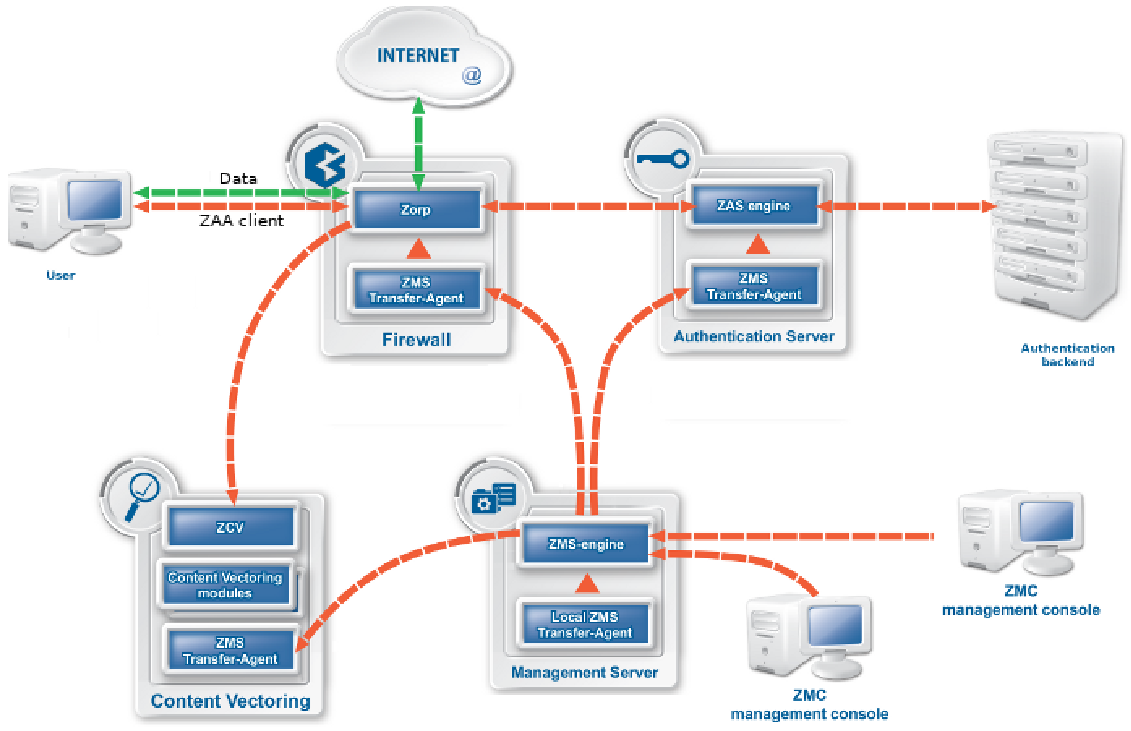


Figure 13.1. Zorp components communicate using agents

For further information on ZMS and the basic architecture, see *Chapter 2, Concepts of the Zorp Gateway solution (p. 3)*.

To modify firewall settings carry out the following procedure regardless of which component is configured.

1. Make the necessary changes in a component's configuration.
   Changes can be undone with the **Revert** option as long as they are not committed to the ZMS database.

2. Commit the new configuration to the ZMS database.

The ZMS host stores the modified information in its XML database. Remember to commit the changes before leaving the component.

View the new configuration and compare it with the current firewall configuration with the help of the **View** and **Check** options, respectively.

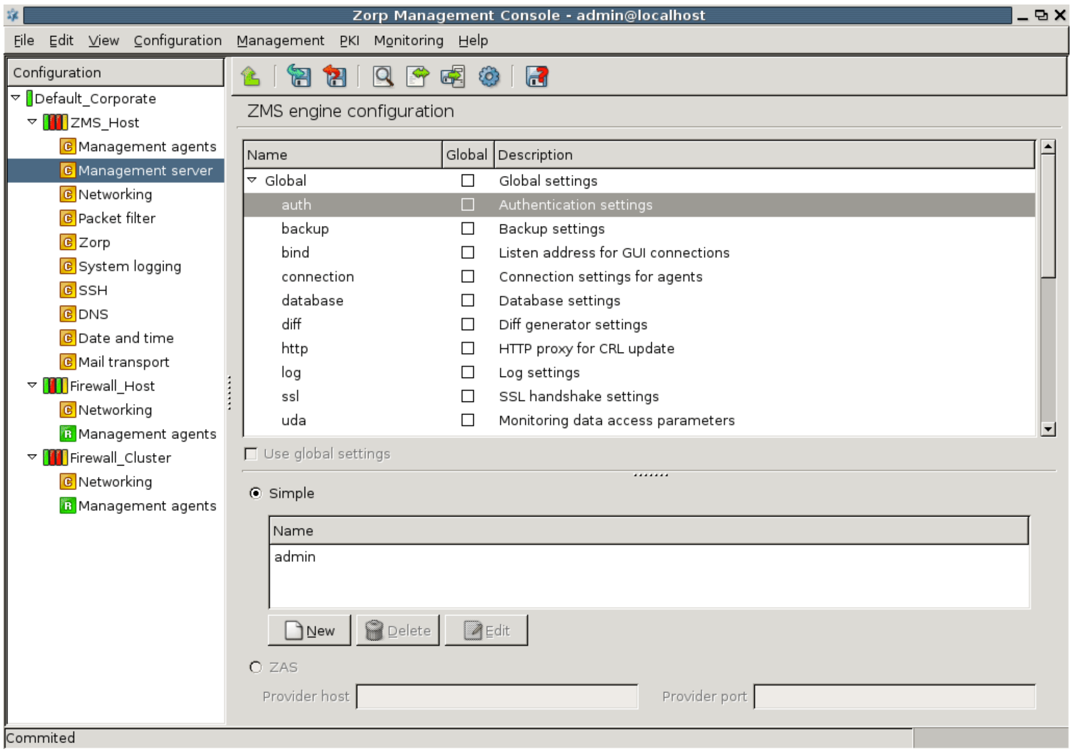
3. Upload the configuration to propagate the changes from the ZMS database down to the firewalls(s). During this process the ZMS converts the configuration data to the proper configuration file format and sends them to the transfer agents on the firewall nodes.

4. Reload the altered configuration or restart the corresponding services to activate the changes. Typically, reloads or restarts are performed after finishing all configuration tasks with the various service components.

For more details, see *Chapter 3, Managing Zorp hosts (p. 15)*.

13.1. Setting configuration parameters

ZMS configuration is realized by setting the appropriate parameters of the **Management server** component.



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

[✓ Simple] ZAS

Name
admin

Figure 13.2. The Management server component

The following parameters can be configured.

Parameter name		Description
`auth`	Authentication settings	handling user data and passwords

Parameter name		Description
`backup`	Backup settings	configuring backup
`bind`	Listen address for GUI connections	setting connection between ZMS and ZMC
`connection`	Connection settings for agents	defining connection parameters
`database`	Database settings	defining saving to disk
`diff`	DIFF generator settings	commanding configuration check
`http`	http proxy for CRL settings	defining proxy address
`log`	Log settings	configuring logging parameters
`ssl`	SSL handshake settings	configuring SSL settings for ZMS and agent connection

Table 13.1. ZMS configuration parameters

By using global settings it is possible to apply default values to the parameter set.

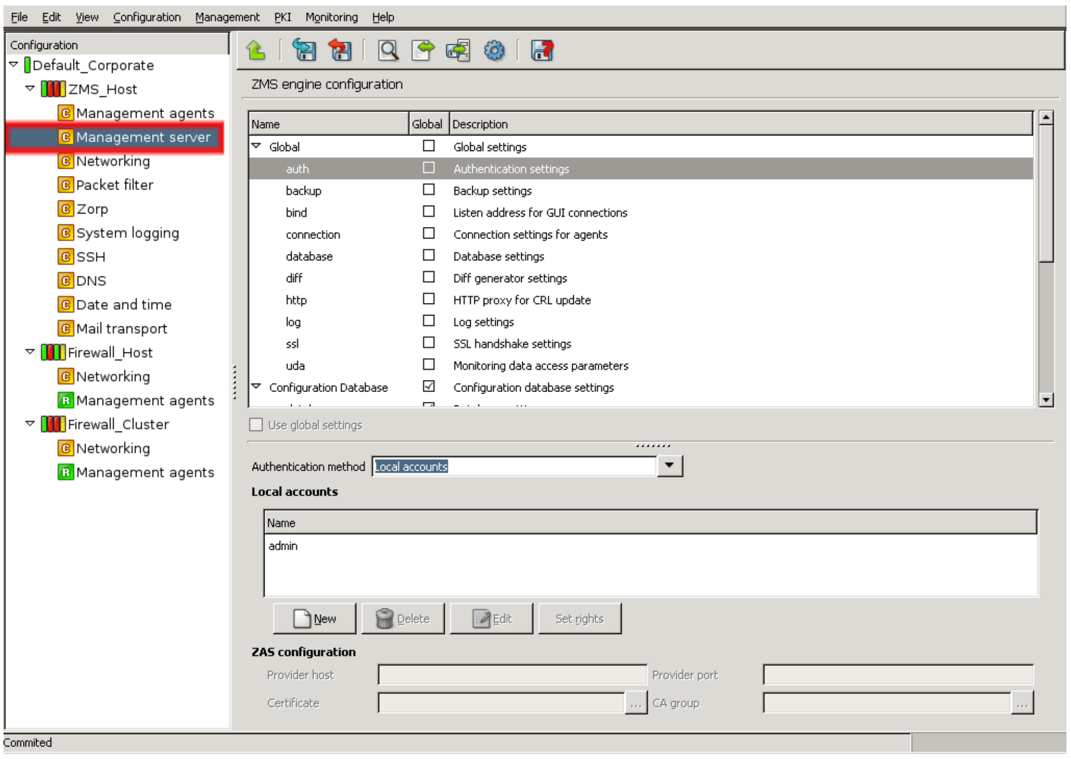
Note
It is recommended to use the global settings when no special configuration is needed.

Different configurations are possible for the following subsystems:

- configuration database,
- key management,
- and transfer engine.

13.1.1. Configuring user authentication and privileges

With the *`Authentication settings (auth)`* parameter the access to ZMS can be configured.



ZMS engine configuration

Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters
Configuration Database	☑	Configuration database settings

☐ Use global settings

Authentication method: Local accounts

Local accounts

Name
admin

ZAS configuration

Provider host: Provider port:

Certificate: CA group:

Figure 13.3. ZMS authentication settings

Users are listed in the main textbox. The following tasks can be performed:

- *Adding new users to ZMS*
- *Deleting users from ZMS*
- *Changing passwords in ZMS*
- *Editing user privileges ZMS*
- *Configuring authentication settings in ZMS*

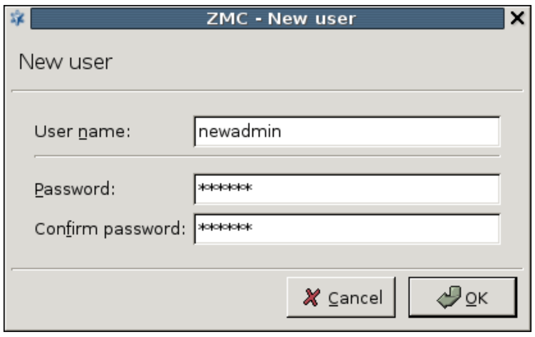
Note
Only the `admin` user can delete users, or modify the password and privileges of another user.

13.1.1.1. Procedure – Adding new users to ZMS

Step 1. Navigate to the **Management server** component of the host running ZMS, and select the **auth** parameter from **Global** parameters.

Step 2. Click **New** to add new users to the system.

Step 3. Enter username and password in the opening window.



New user

User name: newadmin

Password: ******

Confirm password: ******

Figure 13.4. Adding a new ZMS user

Step 4. Confirm password.

Step 5. Click **OK**, commit and upload the changes, and reload the **Management server** component.

13.1.1.2. Procedure – Deleting users from ZMS

Note
Only the `admin` user can delete users, or modify the password and privileges of another user.

Step 1. Navigate to the **Management server** component of the host running ZMS, and select the **auth** parameter from **Global** parameters.

Step 2. Select the user required to be deleted and click **Delete**.

Step 3. Commit and upload the changes, and reload the **Management server** component.

13.1.1.3. Procedure – Changing passwords in ZMS

Note
Only the `admin` user can delete users, or modify the password and privileges of another user.

Step 1. Navigate to the **Management server** component of the host running ZMS, and select the **auth** parameter from **Global** parameters.

Step 2. Select the username whose password is required to be changed.

Step 3. Click **Edit**.

Step 4. Enter the current password and a new password in the opening window.

Step 5. Confirm the new password.

Step 6. Click **OK**, commit and upload the changes, and reload the **Management server** component.

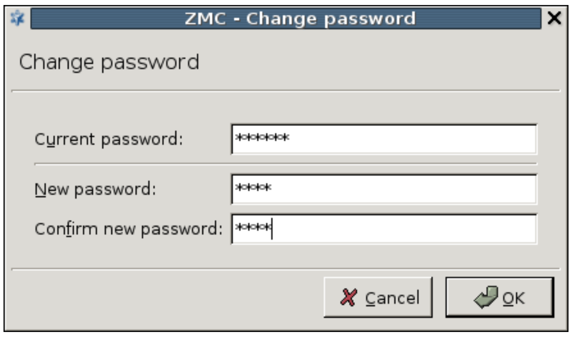


Figure 13.5. Change ZMS user password

13.1.1.4. Configuring user privileges in ZMS

In order to help configuration auditing and the general process of Zorp administration, ZMC users can now have different access rights. That way different administrators can have different responsibilities — PKI management, log analysis, Zorp configuration, and so on.

Note
Only the `admin` user can delete users, or modify the password and privileges of another user.

13.1.1.4.1. Procedure – Editing user privileges in ZMS

Note
Only the `admin` user can delete users, or modify the password and privileges of another user.

Step 1. Navigate to the **Management server** component of the host running ZMS, and select the **auth** parameter from **Global** parameters.

Step 2. Select the username whose privileges are required to be edited.

Step 3. Click **Set rights**.

Note
To change the password of the user, click **Edit**.

Step 4. Select the privileges required to be granted to the user. A user can have none, any, or all of the following privileges:

- *Modify configuration*: Modify and commit the configuration of the hosts. The user can perform any configuration change, and commit them to the ZMS database, but cannot activate the changes or control any services or components.
- *Control services*: Start, stop, reload, or restart any instance, service, or component. This right is required also to upload configuration changes to the hosts.

- *PKI*: Manage the public key infrastructure of Zorp: generate, sign, import and export certificates, CAs, and so on.
- *Log view*: View the logs of the hosts.

To create a 'read-only' user account for auditing purposes, do not select any privileges.

To create a user account with full administrator rights, select every privilege.

Step 5. Click **OK**, commit and upload the changes, and reload the **Management server** component.

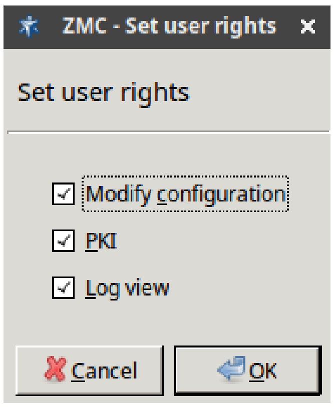


Figure 13.6. Edit user privileges

13.1.1.5. Configuring authentication settings in ZMS

Users connecting to ZMS using ZMC must authenticate themselves. The following authentication methods are available:

- *Local accounts*: ZMS stores the usernames and passwords in a local database. This is the default authentication method.
- *Local accounts and ZAS authentication*: ZMS stores the usernames locally, but receives the authenticattion of users against a ZAS instance. All users successfully authenticating against ZAS and having a local account can connect to ZMS.

13.1.1.5.1. Procedure – Modifying authentication settings

Step 1. Navigate to the **Management server** component of the host running ZMS, and select the **auth** parameter from **Global** parameters.

Step 2. Select the desired authentication method in the **Authentication method** field.

Step 3. If **Local accounts and ZAS authentication** has been selected, configure access to ZAS in the **ZAS configuration** section.

Note
Using these authentication methods requires an already configured ZAS instance. See *Chapter 15, Connection authentication and authorization (p. 389)* for details on using and configuring ZAS.

Enter the IP address or the hostname of the Zorp Authentication Server into the **Provider host** field. By default, ZAS accepts connections on port *`1317`*.

Select the certificate that ZMS will use to authenticate itself from the **Certificate** field.

Select the CA group that contains the CA that issued the certificate of ZAS from the **CA group** field. ZMS will use this group to verify the certificate of ZAS.

Step 4. If more than one authentication backends are run (more than one ZAS instances), create a new router in the **Authentication server** ZMC component that will direct the authentication requests coming from ZMS to the appropriate ZAS instance.
Add a new condition to the router, and enter *`Authentication-Peer`* into the **Variable** field, and *`zms`* into the value field.

For details on configuring ZAS routers, see *Section 15.3.1.2, Configuring routers (p. 399)*.

Note
ZMS sends also the username in the authentication requests. This can be used to direct authentication requests to different ZAS instances based on the username.

Step 5. Click **OK**, commit and upload the changes, and reload the **Management server** component.

13.1.2. Configuring backup

With the *`Backup settings (backup)`* parameter it is possible to define the automatic ZMS database backup method. The automatic backup can be enabled and then the base time for the first backup can be determined, also the interval between all subsequent backup processes can be defined. Additionally, the number of stored database backup copies can be set. Alternatively, it is possible to create scripts to handle the backup tasks.

Backups are stored in the `/var/lib/zms/backup` directory of the ZMS host. The name of the backup file is `zms-backup-<timestamp>.tar.gz`

Warning
If automatic backup is not enabled save the database manually, otherwise all database information might get lost.

13.1.2.1. Procedure – Configuring automatic ZMS database backups

Step 1. Select **Enable automatic backup**.

Step 2. Enter **Base time** for the first backup.
Use hours:minutes format, for example, 08:00.

Step 3. Define the **Interval** between the backups.
Use hours:minutes format, for example, 00:30. In most cases it is sufficient to backup once or twice daily.

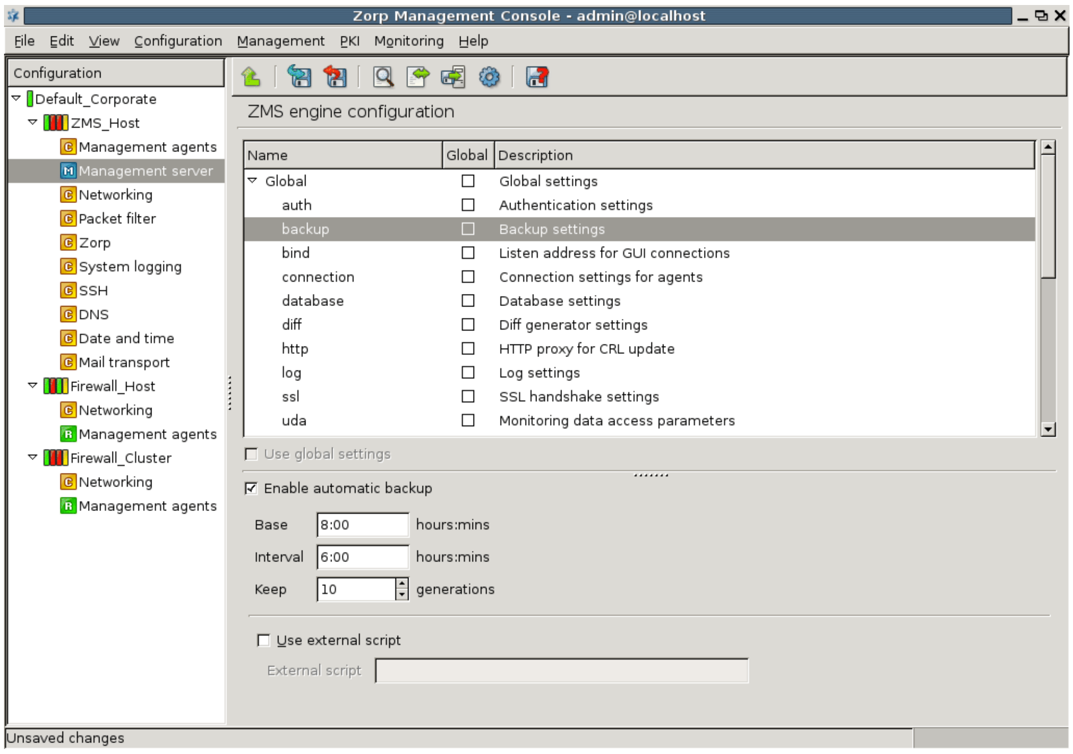
Step 4. Select the number in **Keep generation field** to define how many backup records are stored.

The default value is 10, meaning that only the last 10 database backups are available. The allowed values are ranging from 1 to 100.

The more backups are stored, the more space it reserves. Since one record is only 1-2 MB it is possible to keep 20-50 records if needed without overloading the system.

Step 5. (OPTIONAL)
Tick **Use external script** and give the path of the desired script if a special script for the backup is required to be used.

Tip
Using an external script is a good way to copy the backups to an external host, for example, a backup server.



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

☐ Use global settings

☑ Enable automatic backup

Base	8:00	hours:mins
Interval	6:00	hours:mins
Keep	10	generations

☐ Use external script

External script

Unsaved changes

Figure 13.7. Maintenance of database backup

13.1.2.2. Procedure – Restoring a ZMS database backup

The following steps describe how to restore an earlier ZMS database.

Warning
Restoring an earlier database will delete the current database, including the configuration of every host, as well as any certificates stored in the PKI system. Any modification and configuration change performed since the backup was created will be lost.

Step 1. Copy the database backup archive file to the ZMS host.

Step 2. As root, issue the `/usr/share/zms/zms-restore.sh <backup-file-to-restore>` command.

Step 3. Follow the on-screen instructions of the script.

Step 4. Login to ZMS using ZMC. If ZMS has been reinstalled, use the username and password provided during the reinstallation.

Step 5. If the restored database has to be upgraded, ZMC displays a list of the components to be upgraded. Click **Convert**.

Step 6. Select **PKI** > **Distribute Certificates**. Note that key distribution will fail on every host except on the ZMS host. This is normal.

Step 7. Upload the configuration of the ZMS host.

Step 8. Restart at least the Management Server component on the ZMS host. This will terminate the ZMC session.

Step 9. Relogin with ZMC and check the restored configuration.

Step 10. Upload and restart any component as needed. It may also be needed to redistribute the certificates.

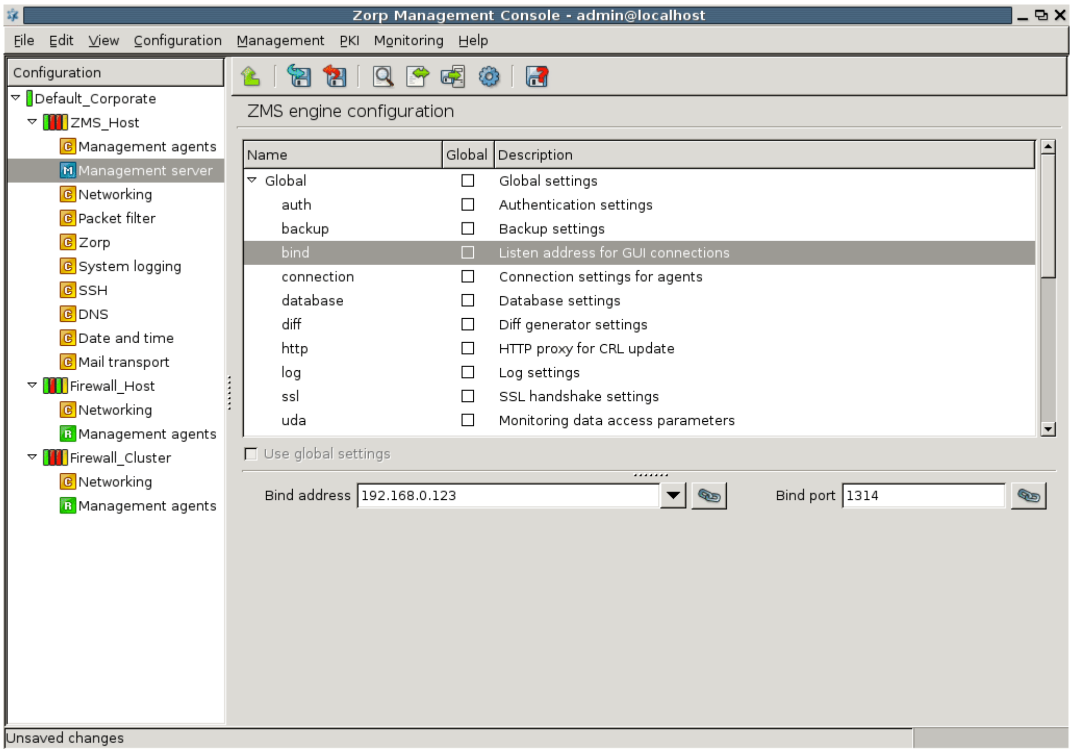
13.1.3. Configuring the connection between ZMS and ZMC

With the *`Listen address for GUI connections (bind)`* parameter it is possible to configure the connection between the ZMS and the ZMC. Give the bind address and the bind port to define where to listen for the GUI connection.

13.1.3.1. Procedure – Configuring the bind address and the port for ZMS-ZMC connections

Step 1. Provide the bind address. The following alternatives are available to define the bind address.

- Enter the IP address manually.



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

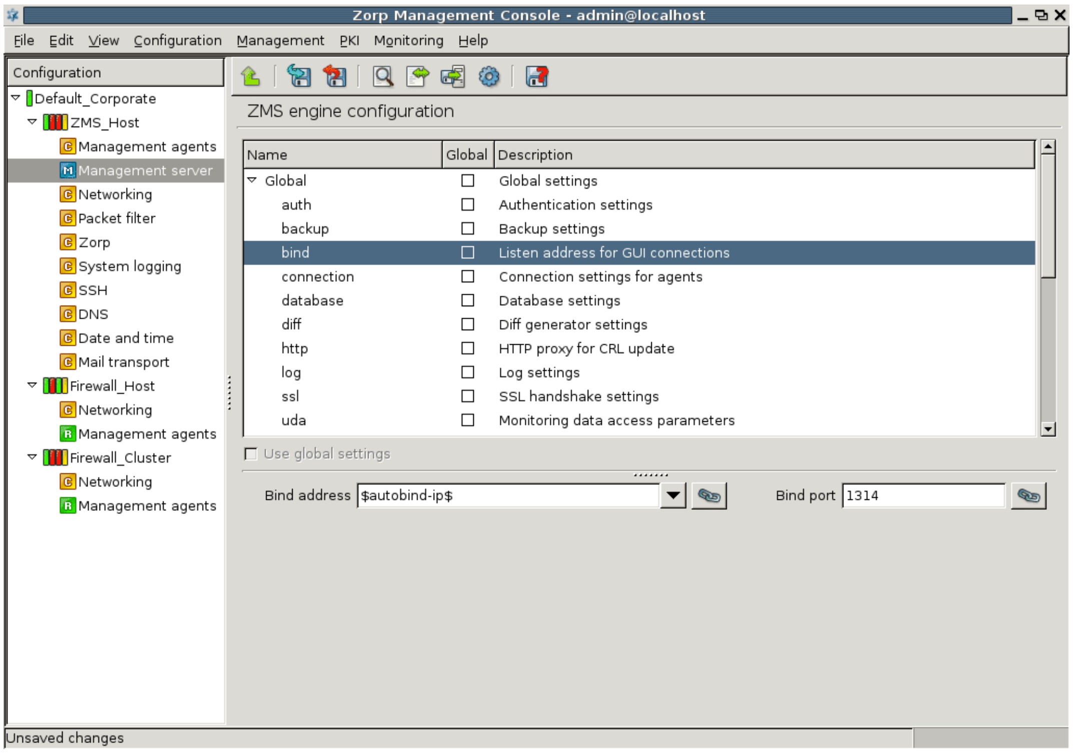
Bind address 192.168.0.123 Bind port 1314

Figure 13.8. Configuring IP address manually

- Select the needed IP address from the drop-down menu.
  The menu shows the available IP interface addresses.

Note
Note that in case the IP address changes for some reason, this data needs to be modified manually since changes are not propagated automatically.

- Use variables.


Zorp Management Console - admin@localhost

File Edit View Configuration Management PKI Monitoring Help

Configuration
- Default_Corporate
  - ZMS_Host
    - Management agents
    - Management server
    - Networking
    - Packet filter
    - Zorp
    - System logging
    - SSH
    - DNS
    - Date and time
    - Mail transport
  - Firewall_Host
    - Networking
    - Management agents
  - Firewall_Cluster
    - Networking
    - Management agents

ZMS engine configuration

Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

☐ Use global settings

Bind address $autobind-ip$ Bind port 1314

Unsaved changes

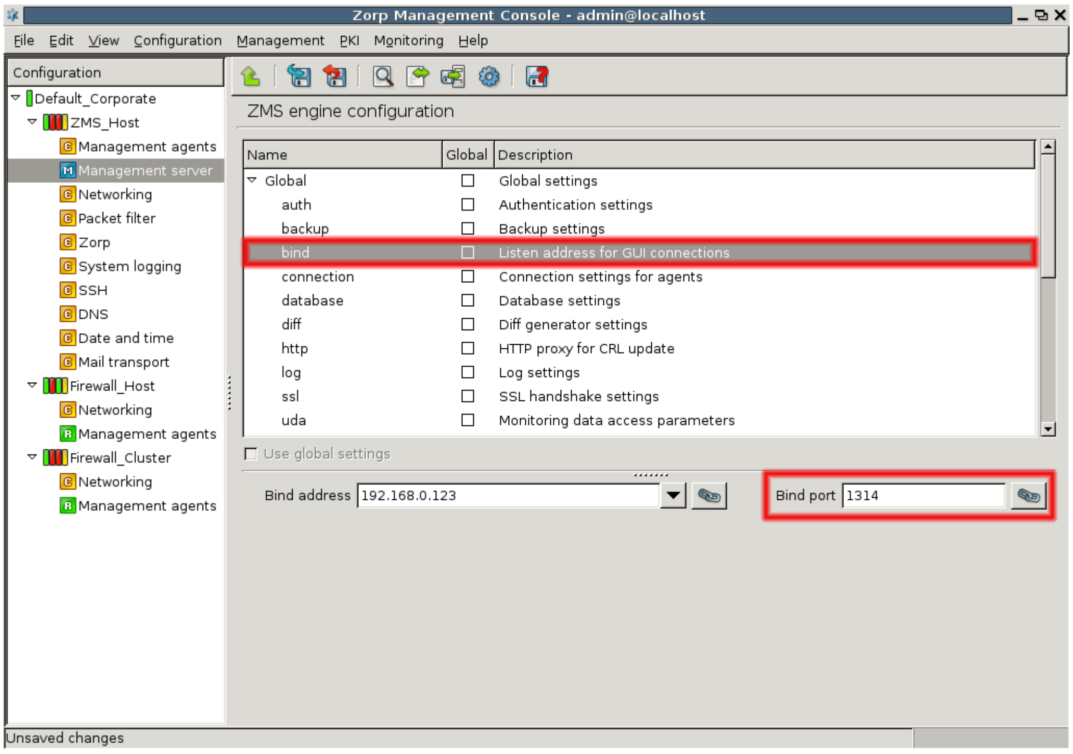
Figure 13.9. Using variables for IP address configuration

ZMS resolves variables during configuration generation (view, check and upload processes). For more information on using variables, see *Chapter 3, Managing Zorp hosts (p. 15)*.

- Create link to an IP address.

Note
It is recommended to give the address this way so that future address changes will have no effects on the operability of the connection.

Step 2. Provide bind port. It is possible to define bind ports similarly to bind address.



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

☐ Use global settings

Bind address 192.168.0.123

Bind port 1314

Figure 13.10. Setting the bind port for ZMC connection

- Enter the port number manually.
- Use variables.
- Create link to a port

13.1.3.2. Procedure – Using linking for the IP address

Step 1. Click the link icon.

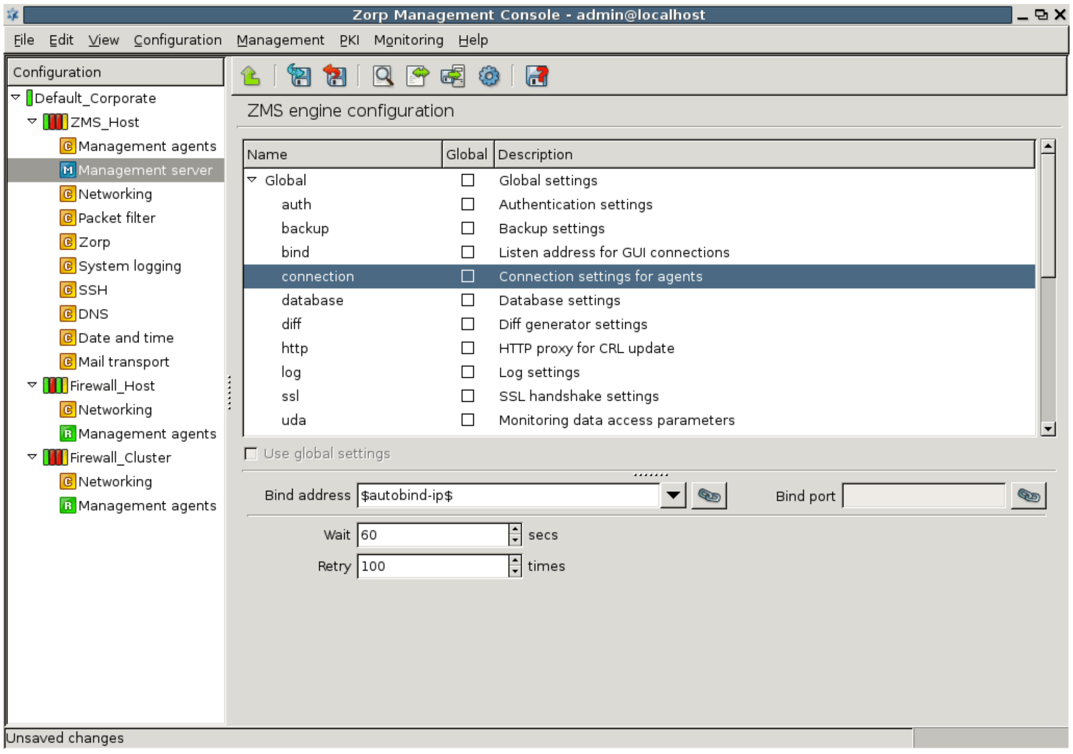
Step 2. Select the link target in the opening window.

Step 3. Click **OK**.
If **Unlink** and **Unlink as value** options are selected, the existing links are deleted. If **Unlink** is chosen it ceases the link connection totally, meaning that the link field is left empty while **Unlink as value** deletes the link but leaves the target IP address in the field which will then behave as a manually added address.

13.1.4. Procedure – Configuring ZMS and agent connections

With the *`Connection settings for agents (connection)`* parameter it is possible to determine which bind address and port are required to be used for the connection between ZMS and the agents. Set the awaiting time and the number of retry if needed.

Defining bind address and ports for the agents is done similarly to defining address and ports between ZMS and ZMC. By default, the ZMS initiates the connection, but if ZMS port and address are provided for the connection parameter at the agent's side, the agent can also establish the connection. In that case, the same port and address shall be set for this parameter too.



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

☐ Use global settings

Bind address $autobind-ip$ Bind port

Wait 60 secs

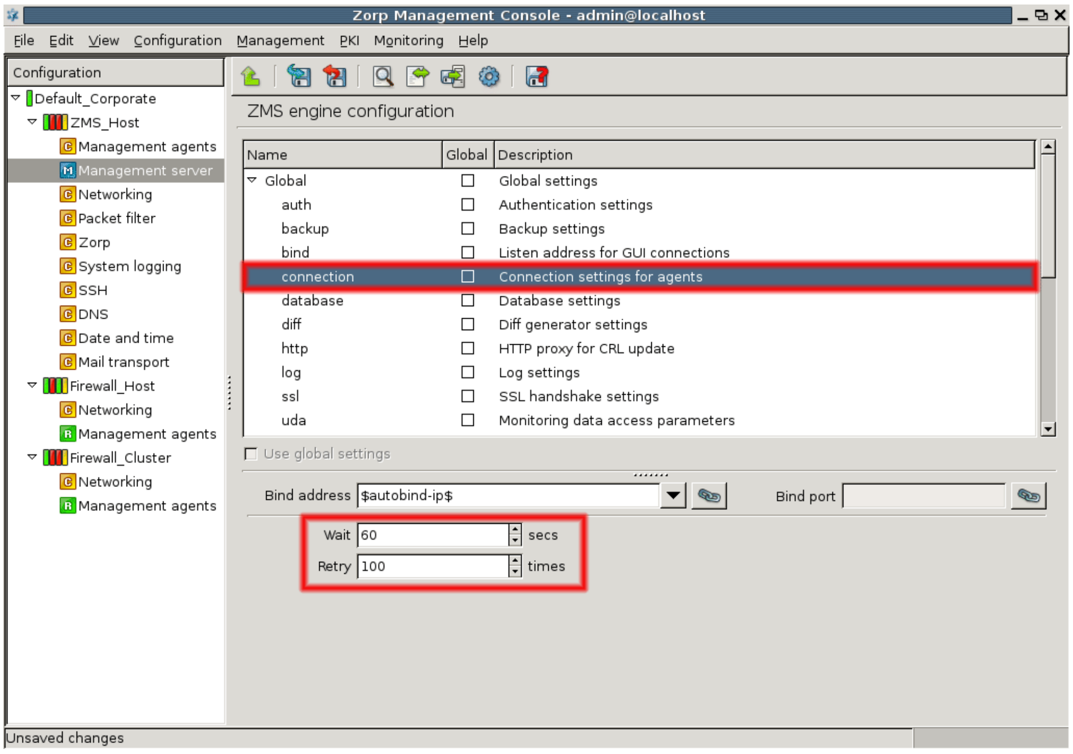
Retry 100 times

Figure 13.11. Agent connection setting

Step 1. Define bind address. The bind address can be defined manually, selected from the available interface addresses, or using variables or links (these possibilities are described in *Section 13.1.3, Configuring the connection between ZMS and ZMC (p. 332)* in detail).

Step 2. Set waiting time and the number of retries. These configure how long ZMS waits for live communication with the agents and in case the connection cannot be set up, how many times ZMS attempts to connect to the agents. Use the **Wait** and **Retry** fields.

Note
If a too low value is defined for the waiting time or a too high value for retry, it is possible to overload the system and cause unnecessary network traffic. Default values are optimal in most cases.



Zorp Management Console - admin@localhost

File Edit View Configuration Management PKI Monitoring Help

Configuration

- Default_Corporate
  - ZMS_Host
    - Management agents
    - Management server
    - Networking
    - Packet filter
    - Zorp
    - System logging
    - SSH
    - DNS
    - Date and time
    - Mail transport
  - Firewall_Host
    - Networking
    - Management agents
  - Firewall_Cluster
    - Networking
    - Management agents

ZMS engine configuration

Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

☐ Use global settings

Bind address $autobind-ip$ Bind port

Wait 60 secs

Retry 100 times

Unsaved changes

Figure 13.12. Setting waiting time and retry times

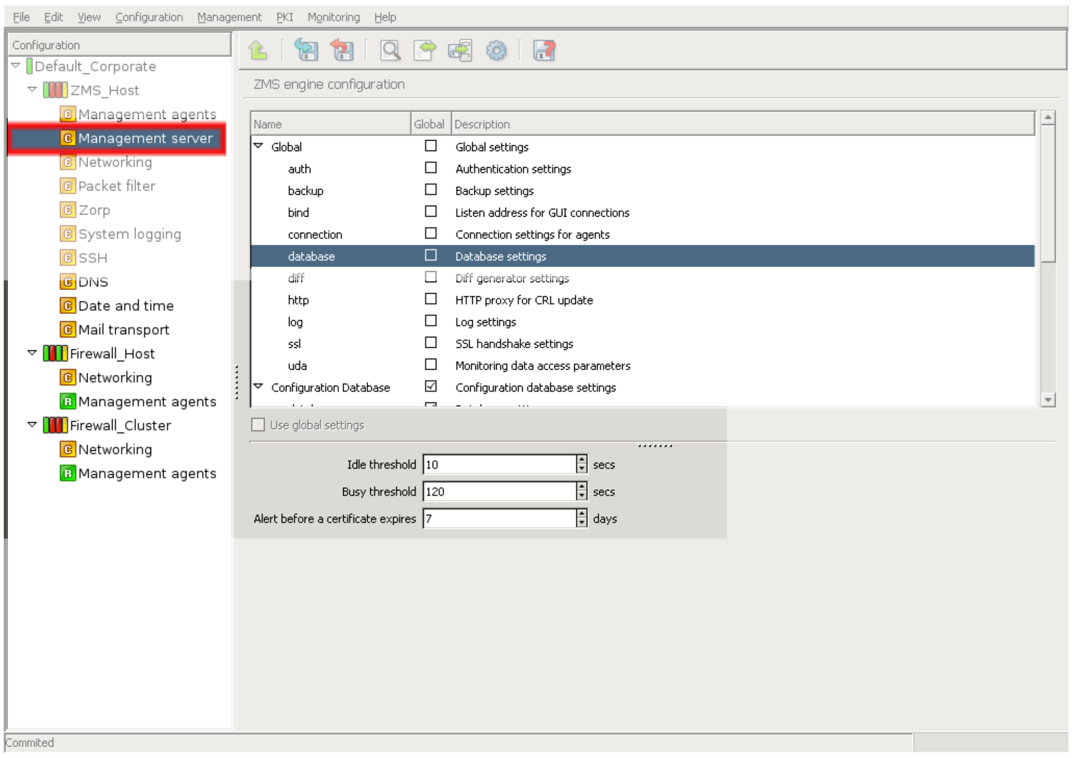
13.1.5. Procedure – Configuring ZMS database save

With the `Database settings (database)` parameter, it can be configured how frequently the ZMS database is saved to disk in both idle and active modes.

Step 1. Select the appropriate number in the **Idle threshold** field to determine the interval between two database backups when the ZMS XML database is not in use, that is, no changes are committed.
Give values in seconds. The default value is 10. All values are allowed.

Step 2. Select the appropriate number in the **Busy threshold** field to determine the interval between two database backups when the ZMS XML database is used and changes are committed.
Give values in seconds. The default value is 120. All values are allowed.

Note
Since the XML database is constantly updated when the ZMS is in use, it is recommended to keep busy threshold value not too low to decrease system load. On the other hand, when choosing high threshold values it is important to bear in mind that during any system breakdown or power outage data might get lost because the database is kept only in the memory and not saved to disk.



- Default_Corporate
  - ZMS_Host
    - Management agents
    - Management server
    - Networking
    - Packet filter
    - Zorp
    - System logging
    - SSH
    - DNS
    - Date and time
    - Mail transport
  - Firewall_Host
    - Networking
    - Management agents
  - Firewall_Cluster
    - Networking
    - Management agents

Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters
Configuration Database	☑	Configuration database settings

☐ Use global settings

Idle threshold	10	secs
Busy threshold	120	secs
Alert before a certificate expires	7	days

Figure 13.13. XML database settings

Step 3. By default, ZMS sends warnings a week before a certificate expires. This value can be modified here.

13.1.6. Setting configuration check

With the *`DIFF generator settings (diff)`* parameter, a command can be defined, to be executed with the **Check configuration** button, when the database on the host is wished to be compared with the ZMS configurations.

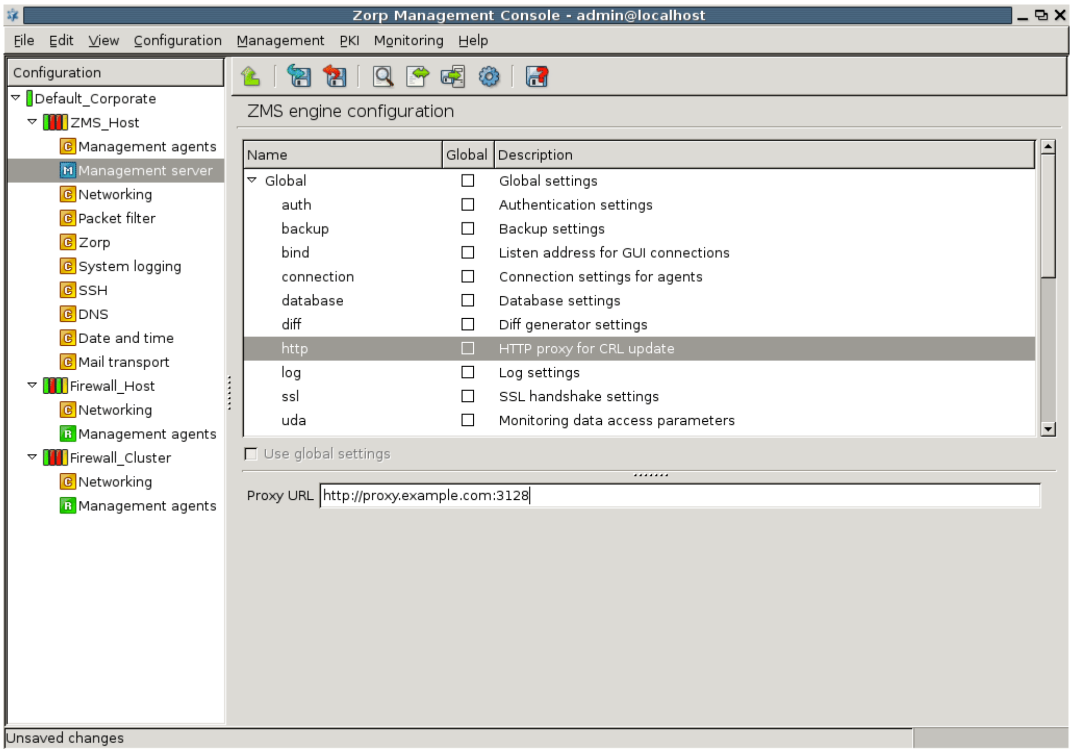
Enter the full path of the command.

The default command is `/usr/bin/diff`. Use special commands, parameters or your own scripts for the configuration check, if needed.

13.1.7. Configuring CRL update settings

With the *`http proxy for CRL settings (http)`* parameter it is possible to give the URL of the proxy required to be used to retrieve the CRL from.

Enter the URL according to the proxy settings, for example, *`http://proxy.example.com:3128/`*.



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

Proxy URL: http://proxy.example.com:3128

Figure 13.14. Setting an HTTP proxy for downloading CRLs

13.1.8. Procedure – Set logging level

With the *`Log settings (log)`* parameter it is possible to determine which type of messages shall be logged. Apply logtags to enable advanced message filtering and also to determine where the messages shall be logged to.

Step 1. Select the desired level of logging.
The default value is level 3 which means that all important events are logged but detailed debug information is not. The higher the value the more information is logged. The levels from 0 to 10 are allowed.

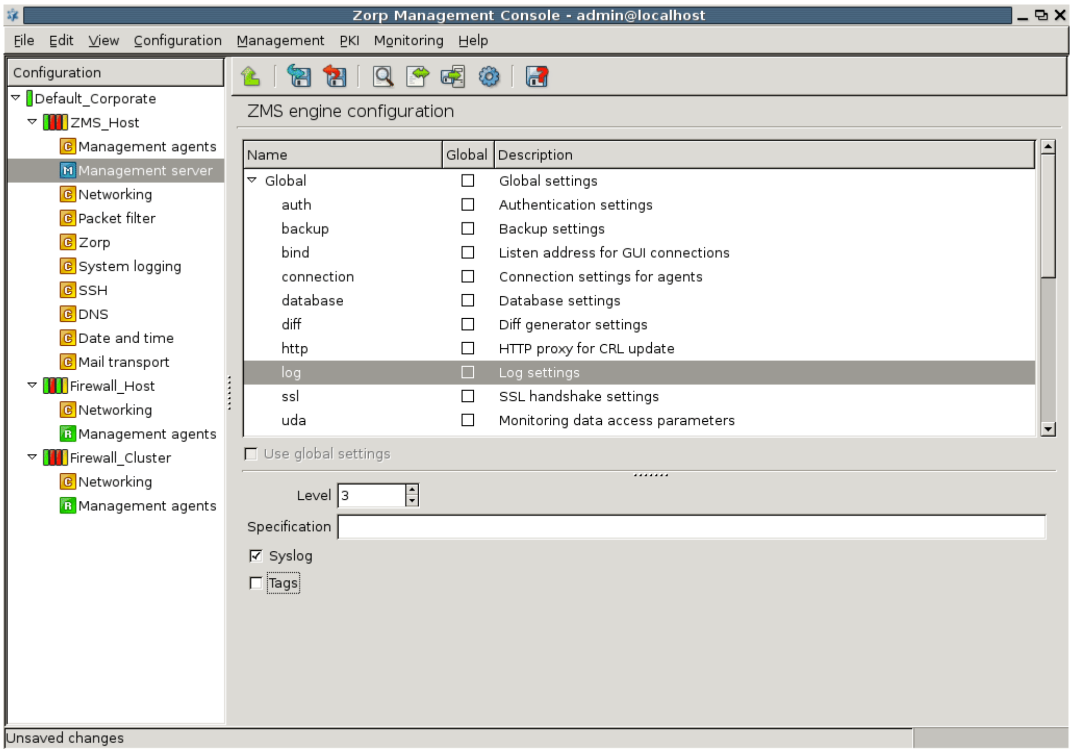
Step 2. (OPTIONAL)
Give log specification for fine-tuning which logtags what level of messages are logged, for example, *`core.debug: 2, session.error:8,*. accounting:5.`*.

Step 3. (OPTIONAL)
Select **Syslog** if syslog is required to be logged, otherwise the messages are logged to the standard output (STDOUT).

Tip
Logging on the console might be useful during troubleshooting.

Step 4. (OPTIONAL)
Select **Tags** if logtags are required to be logged as well.

Tip
This is beneficial if log messages need to be searched and analysed, but it requires more disk space.



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

☐ Use global settings
Level 3
Specification
☑ Syslog
☐ Tags

Figure 13.15. Settings of ZMS logging method

13.1.9. Procedure – Configuring SSL handshake parameters

With the *`SSL handshake settings (SSL)`* parameter the certificate verification parameter and other handshake-related information can be set.

Step 1. Select verification level in the **Verify depths** field to decide how many levels are verified in the certificate hierarchy.
Values from 0 to 100 are allowed.

Step 2. Choose **Groups** or **Advanced** with the radio buttons.

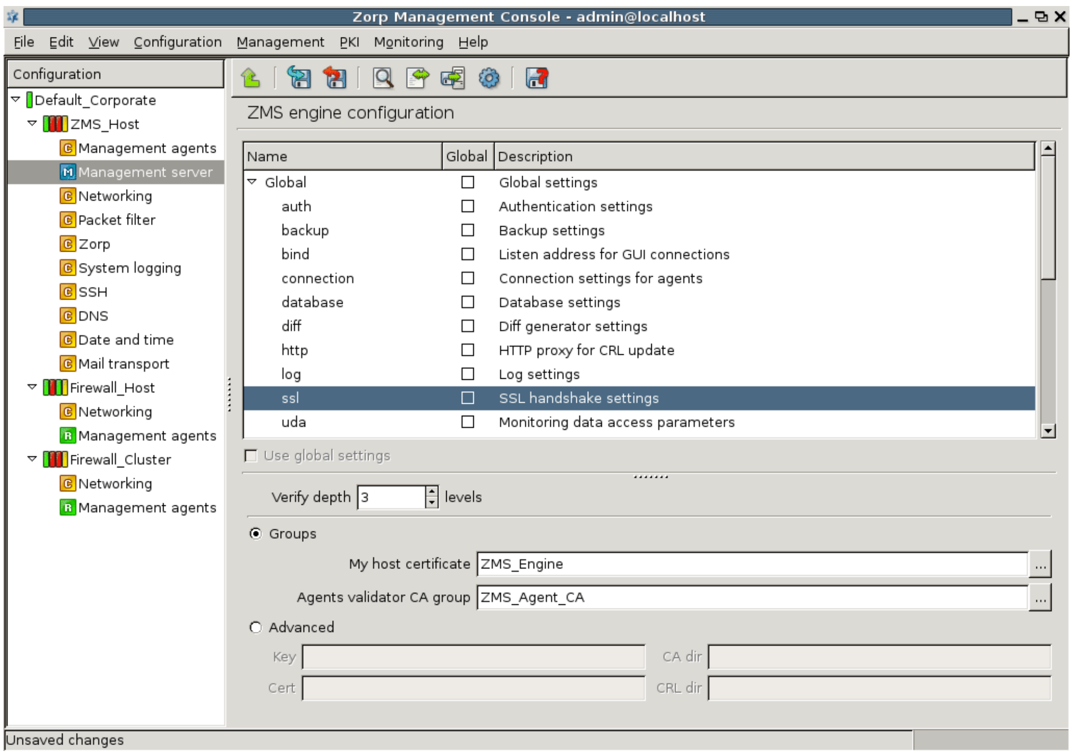
Note
It is recommended to use the PKI groups configuration.

Step a. In **Groups** settings select the certificate entity for the ZMS host.

For example: ZMS_engine: If the Certificate selector window is opened, it displays the unique identifier of the ZMS host and also certificate information, such as version, serial number, issue date and validity period, algorithms and keys. This information is useful when selecting which certificate to use.

Step b. Select agents validator CA group.

For example: ZMS_Host_CA: If the CA group selector window is opened, the CA group can be defined which is used to verify the certificate of the agents during the handshake. Data is available on CA group name, certificate name and certificate information for the selected CA groups.



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

☐ Use global settings

Verify depth 3 levels

[✓ Groups]

My host certificate ZMS_Engine

Agents validator CA group ZMS_Agent_CA

Advanced

Key

CA dir

Cert

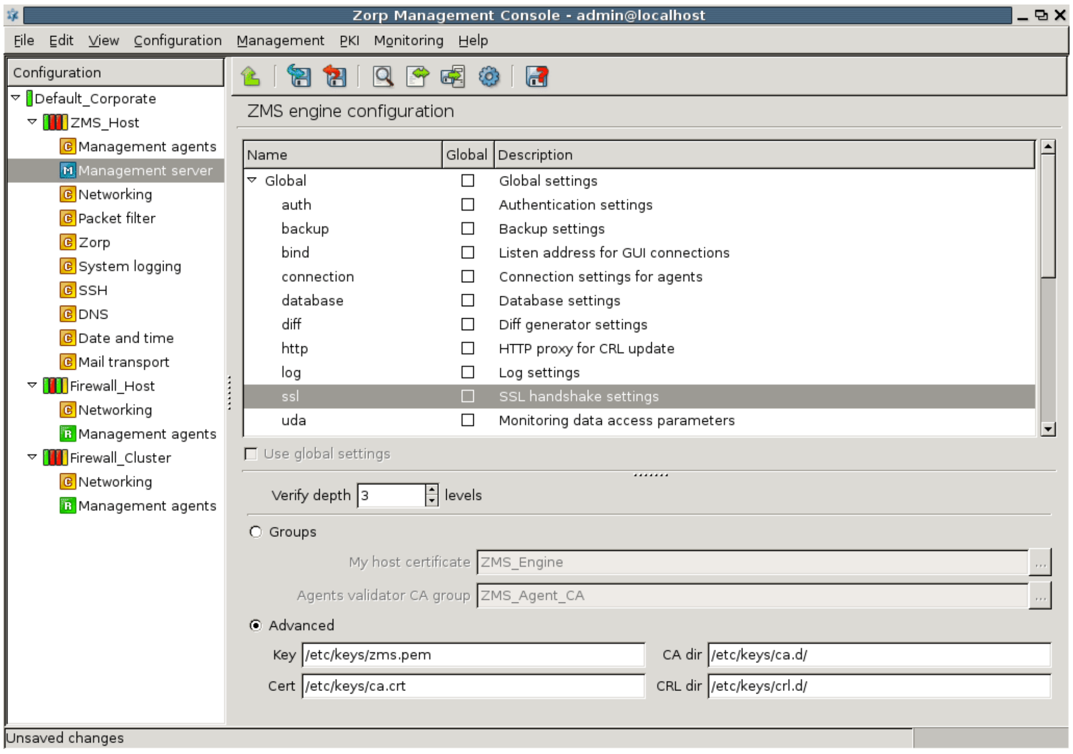
CRL dir

Figure 13.16. SSL settings

OR

Step c. In **Advanced** settings enter manually the following data.

- full path of the file where the private key is stored
- certificate
- CA directory identifying the directory where the CA certificate entities are stored
- and CRL directory giving the place of the CRLs corresponding to the CA screenshot



Name	Global	Description
Global	☐	Global settings
auth	☐	Authentication settings
backup	☐	Backup settings
bind	☐	Listen address for GUI connections
connection	☐	Connection settings for agents
database	☐	Database settings
diff	☐	Diff generator settings
http	☐	HTTP proxy for CRL update
log	☐	Log settings
ssl	☐	SSL handshake settings
uda	☐	Monitoring data access parameters

☐ Use global settings

Verify depth 3 levels

Groups

My host certificate ZMS_Engine

Agents validator CA group ZMS_Agent_CA

[✓ Advanced]

Key /etc/keys/zms.pem CA dir /etc/keys/ca.d/

Cert /etc/keys/ca.crt CRL dir /etc/keys/crl.d/

Figure 13.17. Advanced settings for SSL connection

13.2. Setting agent configuration parameters

Agent configuration, similarly to ZMS configuration, is realized by setting the appropriate parameters. The following parameters can be configured for the agents.

Parameter name		Description
`connection`	Connection settings for agents	defining connection parameters
`engine`	Engines to connect	configuring connection to engines
`log`	Log settings	configuring logging parameters
`ssl`	SSL handshake settings	configuring SSL settings for ZMS and agent connection

Table 13.2. Agent configuration parameters.

By using global settings it is possible to apply default values to the parameter set.

Note
It is recommended to use the global settings when no special configuration is needed.

13.2.1. Configuring connections for agents

With the `Connection settings (connection)` parameter it can be set which bind address and port is required to be used for the connection between the agent and the ZMS. The agent receives or initiates the connection from / to the ZMS engine using this address. The waiting time and the number of retries can also be set. Note that it is recommended to allow ZMS to establish the connection towards the agent. The configuration process is identical to the one described in *Section 13.1.3, Configuring the connection between ZMS and ZMC (p. 332)*.

For further information, see *Section 13.2.1, Configuring connections for agents (p. 343)* and *Section 13.3, Managing connections (p. 346)*.

13.2.2. Configuring connection to engine

With the `Engines to connect (engine)` parameter it can be configured to which engine the agent connects to. Generally, it is the engine that connects to the agent but if this parameter is set, the agent initiates the connection based on these settings.

Note
It is recommended to leave this parameter empty and allow the ZMS engine to establish the connection towards the agents.

13.2.3. Procedure – Configuring logging for agents

With the `Log settings (log)` parameter it can be defined which type of messages should be logged at the agent's side. Logtags can be applied to enable advanced message filtering and also determine where the messages shall be logged to.

Step 1. Select the desired level of logging.
The default value is level 3 which means that all important events are logged but no detailed debug information. The higher the value the more information is logged. The allowed levels are from 0 to 10.

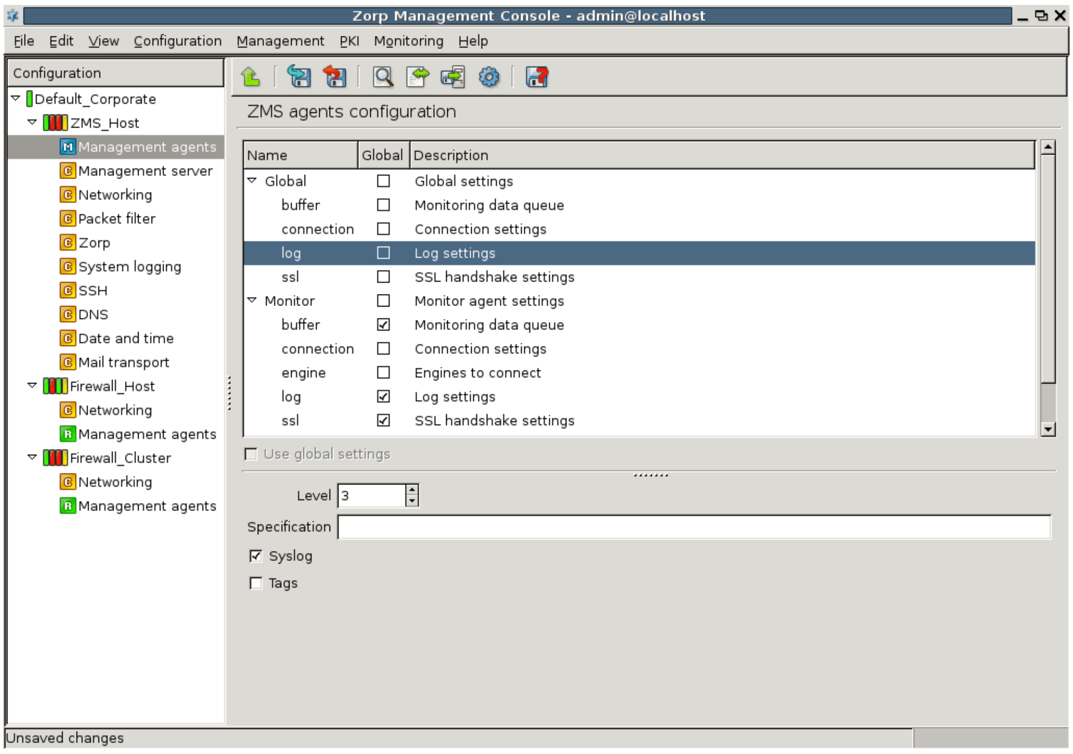
Step 2. (OPTIONAL)
Give log specification if it is required to fine-tune for which logtags which level of messages are logged, for example, `core.debug: 2, session.error: 8,*. accounting:5.`.

Step 3. (OPTIONAL) Tick **Syslog** to log to syslog, otherwise the messages are logged to the standard output (STDOUT).

Tip
Logging on the console might be useful during troubleshooting.

Step 4. (OPTIONAL)
Tick **Tags** to log the logtags as well.

Tip
This is beneficial if log messages need to be searched and analysed, but it requires more disk space.


Zorp Management Console - admin@localhost
File Edit View Configuration Management PKI Monitoring Help

Configuration
- Default_Corporate
  - ZMS_Host
    - Management agents
    - Management server
    - Networking
    - Packet filter
    - Zorp
    - System logging
    - SSH
    - DNS
    - Date and time
    - Mail transport
  - Firewall_Host
    - Networking
    - Management agents
  - Firewall_Cluster
    - Networking
    - Management agents

ZMS agents configuration

Name	Global	Description
Global	☐	Global settings
buffer	☐	Monitoring data queue
connection	☐	Connection settings
log	☐	Log settings
ssl	☐	SSL handshake settings
Monitor	☐	Monitor agent settings
buffer	☑	Monitoring data queue
connection	☐	Connection settings
engine	☐	Engines to connect
log	☑	Log settings
ssl	☑	SSL handshake settings

☐ Use global settings

Level 3

Specification

☑ Syslog

☐ Tags

Unsaved changes

Figure 13.18. Setting up agent logging

13.2.4. Procedure – Configuring SSL handshake parameters for agents

With the *SSL handshake settings (SSL)* parameter the certificate verification parameters for the agent and other handshake-related information can be set to be used between the agent and the ZMS.

Step 1. Select verification level in the **Verify depths** field to decide how many levels are verified in the certificate hierarchy.
Values from 0 to 100 are allowed.

Step 2. Choose **Groups** or **Advanced** with the radio buttons.

Note
It is recommended to use the PKI groups configuration.

Step a. In **Groups** settings select the certificate entity for the agent.
For example: ZMS_engine.

If the **Certificate selector** window is opened, it displays the unique identifier of the ZMS host and also certificate information, such as version, serial number, issue date and validity period, algorithms and keys.

Tip
This information is useful when selecting which certificate to use.

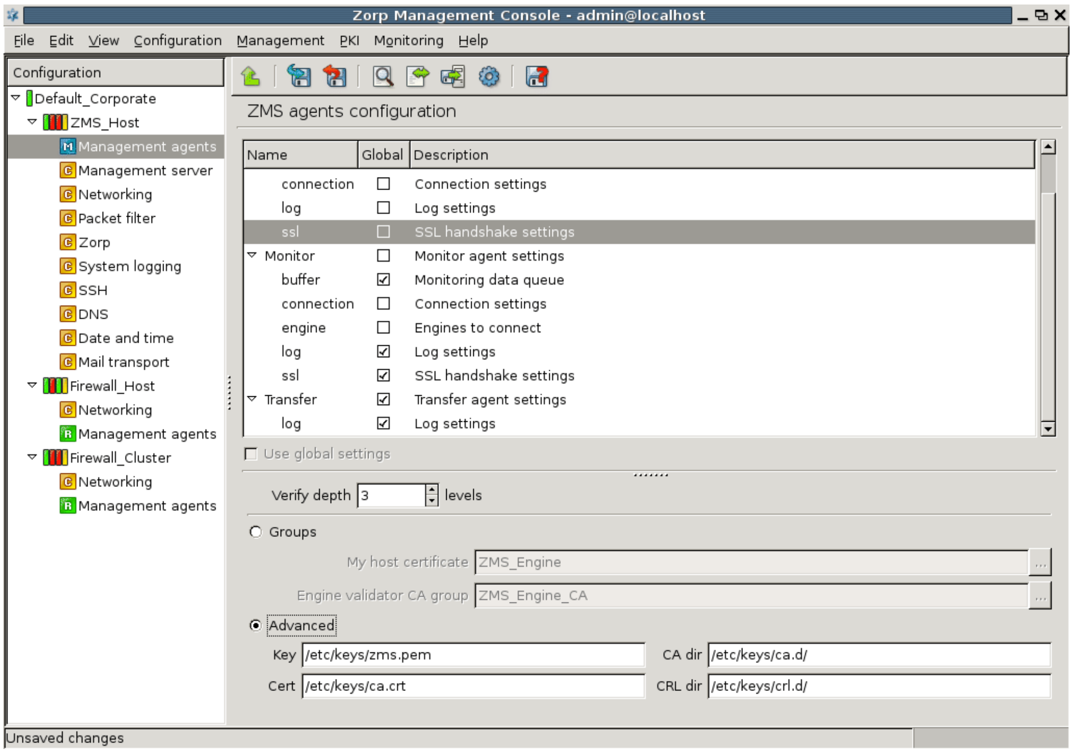
Step b. Select engine validator CA group.
For example: ZMS_engine_CA.

If the **CA group selector** window is opened, the CA group can be defined which is used to verify the certificate of the agents during the handshake. Data is available on CA group name, certificate name and certificate information for the selected CA groups.

OR

Step c. In **Advanced** settings enter manually the following data:

- full path of the file where the private key is stored
- certificate
- CA directory identifying the directory where the CA certificate entities are stored
- CRL directory giving the place of the CRLs corresponding to the CA screenshot


Zorp Management Console - admin@localhost
File Edit View Configuration Management PKI Monitoring Help
Configuration

- Default_Corporate
  - ZMS_Host
    - Management agents
    - Management server
    - Networking
    - Packet filter
    - Zorp
    - System logging
    - SSH
    - DNS
    - Date and time
    - Mail transport
  - Firewall_Host
    - Networking
    - Management agents
  - Firewall_Cluster
    - Networking
    - Management agents

ZMS agents configuration

Name	Global	Description
connection	☐	Connection settings
log	☐	Log settings
ssl	☐	SSL handshake settings
Monitor	☐	Monitor agent settings
buffer	☑	Monitoring data queue
connection	☐	Connection settings
engine	☐	Engines to connect
log	☑	Log settings
ssl	☑	SSL handshake settings
Transfer	☑	Transfer agent settings
log	☑	Log settings

☐ Use global settings

Verify depth 3 levels

○ Groups

My host certificate ZMS_Engine

Engine validator CA group ZMS_Engine_CA

● Advanced

Key	/etc/keys/zms.pem	CA dir	/etc/keys/ca.d/
Cert	/etc/keys/ca.crt	CRL dir	/etc/keys/crl.d/

Unsaved changes

Figure 13.19. Advanced settings of SSL connection parameters

13.3. Managing connections

ZMS communicates with the Zorp firewall software through agents. The `zms-transfer-agent` communicates using TCP port 1311.

The initial connection between the firewall and ZMS needs manual set up, but all subsequent communication channels are established automatically. By default, the ZMS host initiates the connection towards the agents but agents can also establish the communication if configured to do so. It is possible to administer the connections through the ZMS host main workspace and with the help of the **Management / Connections** menu.

13.3.1. Setting up initial connection with management agents

See *Procedure 13.3.4, Configuring recovery connections (p. 348)*.

13.3.2. Configuring connection with agents

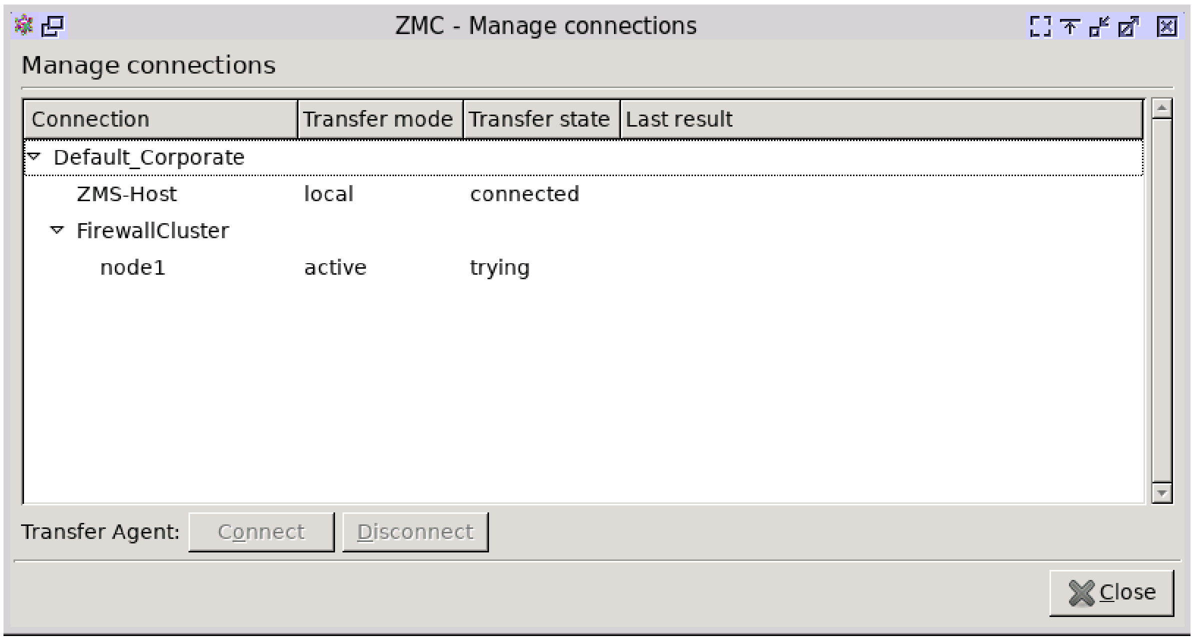
Provide the ZMS port and address for the *`Connection settings for agents (connection)`* parameter, if the agents are required to initiate the connection towards the ZMS using this port and address. The same port and address should be given at the agent's side. If no values are provided, the default connection setup is carried out, that is, the ZMS connects to the agents using the given port and address.

For further information, see chapter *Section 13.2.1, Configuring connections for agents (p. 343)*.

13.3.3. Procedure – Administering connections

Step 1. Query the connection status by choosing the **Management** > **Connections...** menu.
The **Last result** field shows the outcome of the previous connect or disconnect operation.

Step 2. (OPTIONAL)
Stop or Start the communication by selecting the connection and clicking **Connect** or **Disconnect**.



Connection	Transfer mode	Transfer state	Last result
Default_Corporate			
ZMS-Host	local	connected	
FirewallCluster			
node1	active	trying	

Figure 13.20. The managing connection window

The table below briefly describes the values of the **Transfer mode** field.

Transfer mode value	**Description**
`active`	The ZMS initiates the connection towards the agents.
`passive`	The agent establishes the connection.
`disabled`	The connection is not enabled, that is, the connection was disconnected on **Manage connections** window.
`local`	The agent is installed locally on the ZMS host.

Table 13.3. Values of the Transfer mode field and their description

The table below briefly describes the values of the **Transfer state** field.

Transfer state value	**Description**
`disabled`	The agent and the ZMS are disconnected even if the ZMS restarts.

Transfer state value	Description
`connection lost`	The agent and the ZMS are unintentionally disconnected due to an unknown reason.
`trying`	The ZMS is trying to establish the connection.
`listening`	The ZMS is waiting for the agent to establish the connection.
`connected`	The connection is established between the ZMS and the agent.
`unknown`	Unknown state.

Table 13.4. Values of the Transfer state field and their description

13.3.4. Procedure – Configuring recovery connections

Configure a recovery connection in the following cases:

- Connecting a new machine (firewall node) to the ZMS without bootstrapping (to set up the initial connection between ZMS and the Zorp firewall).
- Reconnecting an existing host in case the connection is lost (for example, the used certificate is expired) and the communication cannot be started by using the **Manage Connection** window.
- Installing a new firewall machine to replace a previous one and configuring it based on ZMS data.

The authentication in this case is done using a One-Time-Password (OTP) instead of certificates. After successful authentication, the ZMS receives the configuration data of the agent together with the necessary PKI information (certificate, key and CRL). All further authentication procedures will use this data. After the agent is restarted, the ZMS initiates the reconnection. The administration can be done as normal afterwards.

Note
The agent needs to be in OTP mode to be able to receive the connection.

Note
Passive host temporarily changes to active mode as the agent runs in recovery mode. If the host is behind SNAT without the corresponding DNAT then the recovery will fail.

Step 1. Login to the Zorp host that you want to reconnect to ZMS.

Step 2. Reconfigure the zms-transfer-agent with the following terminal command:`dpkg-reconfigure zms-transfer-agent-dynamic`

Step 3. Enter a One-Time-Password (OTP) that the host will use to connect to ZMS into the window displayed. Enter a password, and store it temporarily for later use.

Step 4. Login to your Zorp Management Server using ZMC.

Step 5. Select the host that needs the recovery connection in ZMC, and click **Recovery connection**.

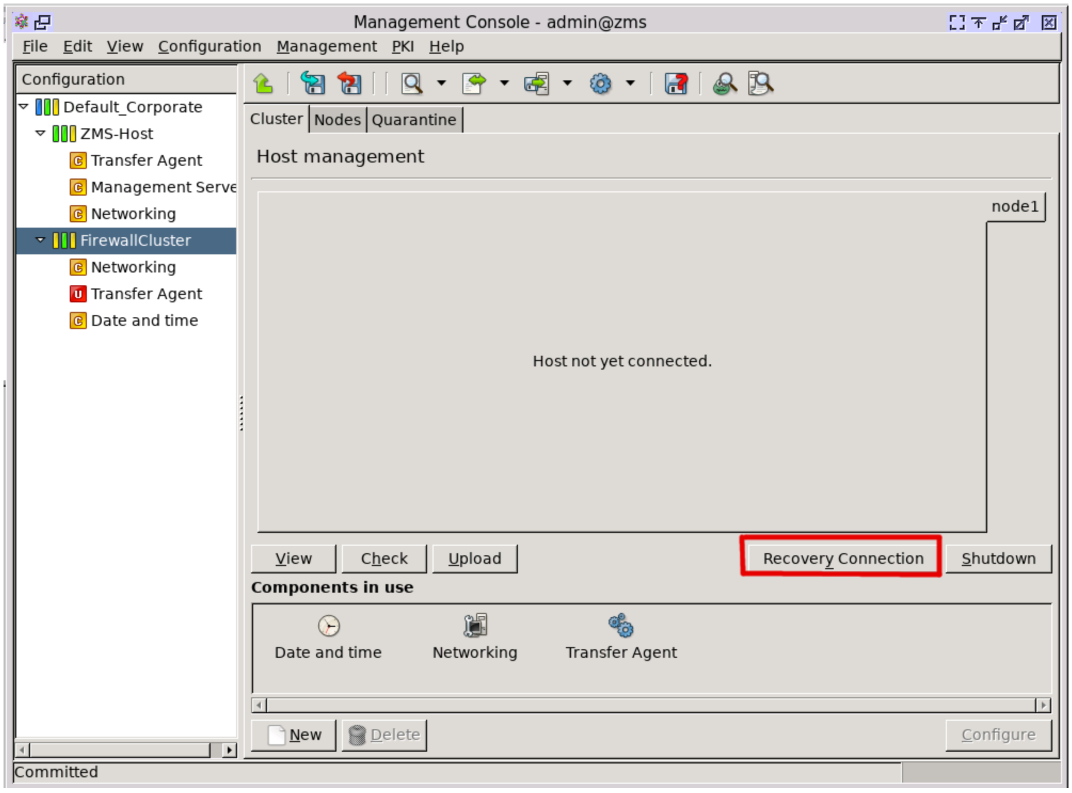


Figure 13.21. Starting a recovery connection

Step 6. Enter the same One-Time-Password (OTP) set during the installation on the host.

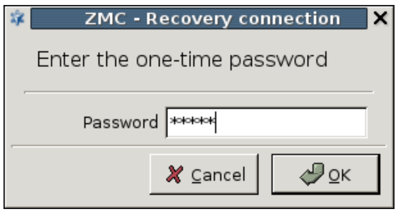


Figure 13.22. Entering the one-time-password

Step 7. Test the connection, for example, stop and start the communication on the **Manage Connection** window or check the system statistics of the **Host** component.

13.4. Handling XML databases

The XML database in the ZMS is functionally divided into two parts and stores two basic types of information.

- predefined information
  For example: proxies, packet filtering parameters or add-ons. This information is provided by Balasys but other definitions can also be freely added.

- configuration settings of your sites, hosts or components
  These are the settings created in ZMC. The configuration files are generated on the basis of this data.

ZMS loads the database information from the `/var/lib/zms` library and places it in the appropriate part of the XML database. During database save ZMS carries out the reverse process: takes the data from the XML database and saves it in the appropriate file and folder of the library.

The `/var/lib/zms` is library is composed of the following folders and XML files.

- `zms_userdatabase.xml`
  including the configuration settings created in the ZMC except for PKI information

- `zms_keymngt.xml`
  including all PKI information and related user settings

- `configdb` folder
  storing templates, databases, definitions necessary for the ZMS such as built-in proxies or other settings provided by Balasys (configuration settings are excluded)

  Note
  Do not change the files in this folder because during upgrade the content of the folder is automatically overwritten. Necessary modifications should be stored in the `configdb-user` folder.

- `configdb-user` folder
  containing the additional settings and templates for the ZMS in separate XML files

  (These files are modified versions of the `configdb/zms_database.xml` file). This data is not deleted during upgrade.

- `keymngt` folder
  containing certificate entities (certificate, key and CRL files generated by PKI)

  Note
  Do not modify this folder.

- `backup` folder
  storing by default the backup of the XML files and folders

  For further information, see chapter *Section 13.1.2, Configuring backup (p. 330)*.

Chapter 14. Virus and content filtering using ZCV

Virus, spam and other types of content filtering services are nowadays essential components of security solutions both in home and in production environments. Spam, viruses, trojans, malicious scripts pose a significant threat to the users through e-mails, downloadable files, or even simple webpages. Firewalls are a logical location for content filtering, as all traffic must travel across the firewall, and usually this is the earliest point where the incoming traffic can be examined and interacted with. ZCV provides an integrated framework to manage the various available content vectoring components using a single, uniform interface.

The sections of this chapter provide an in-depth description of ZCV. For a basic overview of the ZCV framework, content vectoring and a list of the supported modules, see *Section 2.1.6, The concept of the ZCV framework (p. 6)*.

14.1. Content vectoring basics

Content vectoring is basically the act of inspecting the transmitted data (downloaded by a web browser, sent over SMTP, and so on) to detect and reject unwanted content. Depending on the environment and the circumstances, unwanted content can mean viruses, ad- or malware, spam e-mails, client-side scripts (that is, Java, JavaScript, and so on), or simply websites containing information not permitted for the users (such as adult sites, and so on).

Note
Content vectoring may seem to be similar to application level protocol inspection performed by Zorp proxies. However, it is essentially different: the proxies only analyze the elements of the protocol, not the transferred data itself.

The main types of content filtering are summarized below.

- *Virus filtering*: The most classical and well-known form of content vectoring is virus filtering: examining the files being transferred to verify that they do not contain any software that may harm the user's computer or infrastructure. Most virus filtering engines also detect adware and trojan programs. If a virus is detected, often it is possible to remove the virus (disinfect the file) without any side-effect.
- *Spam filtering*: Spam filtering examines e-mails (usually in the SMTP traffic) to delete unwanted advertisements, viruses spreading through e-mails, and so on.
- *Disabling client-side scripts in HTML*: Client-side scripting is a popular method for decreasing the load of webservers. It means that certain actions are performed on the client machine (for example, in a submission form a client-side script could check that all fields are filled, without having to connect to the server). However, such scripts can be exploited to perform virtually any operation on the client machine. Therefore, often they are disabled and completely removed from the webpages as they are downloaded.
- *General HTML content filtering*: Access to certain webpages is also often limited based on the contents of the page — usually based on the keywords occurring in the page. Most commonly this takes the form of blacklisting/whitelisting, to deny access to pages containing prohibited or illegal content, or simply to pages not related to the everyday work of the organization.

Content vectoring is possible using two approaches: file-based and stream-based filtering. File-based filtering is used when the complete object (file) is needed to perform the checking, such as in virus and spam filtering. (Virus filters cannot work on partial files.) Stream-based filtering monitors a continuous data flow (that is, a webpage being downloaded) and removes the prohibited contents (such as JavaScripts, images, and so on).

14.1.1. Quarantining

Objects (infected files, spam e-mails) rejected by the content vectoring system are usually deleted. However, in some environments this is not acceptable: these objects might be temporarily stored in a location where they can do no harm (that is, in the quarantine), until it can be verified that they do not contain any useful information. The reason for using a quarantine is that occasionally information in a file might be needed even though the file is infected (disinfecting a file is not always possible, and sometimes may damage the non-infected parts of the file as well). Also, virus and spam filters are not unerring; occasionally they might detect a file/e-mail as infected/spam even when it is not. If rejected objects are simply deleted, important information might be lost in these cases.

Tip
In production environments it is recommended to use multiple content filtering engines to examine the same traffic to reliably detect viruses. Zorp ZCV fully supports the use of multiple content vectoring engines to inspect the same content.

14.2. Content vectoring with ZCV

This section describes how to initialize and configure ZCV. It is also described how rule groups, scanpaths, and other objects can be created and used to form content vectoring policies. Setting up the communication between ZCV and Zorp is described in *Section 14.2.4, Configuring Zorp proxies to use ZCV (p. 375)*. Before starting to configure ZCV, add the **Content Vectoring** component to the host (see *Procedure 3.2.1.3.1, Adding new configuration components to host (p. 22)* for details).

14.2.1. Creating module instances

Module instances are the elements of the available ZCV modules configured for a particular content vectoring task. A data stream or an object can be inspected by one or more module instances. Module instances can be created and configured on the **Modules** tab of the **Content Vectoring** ZMC component.

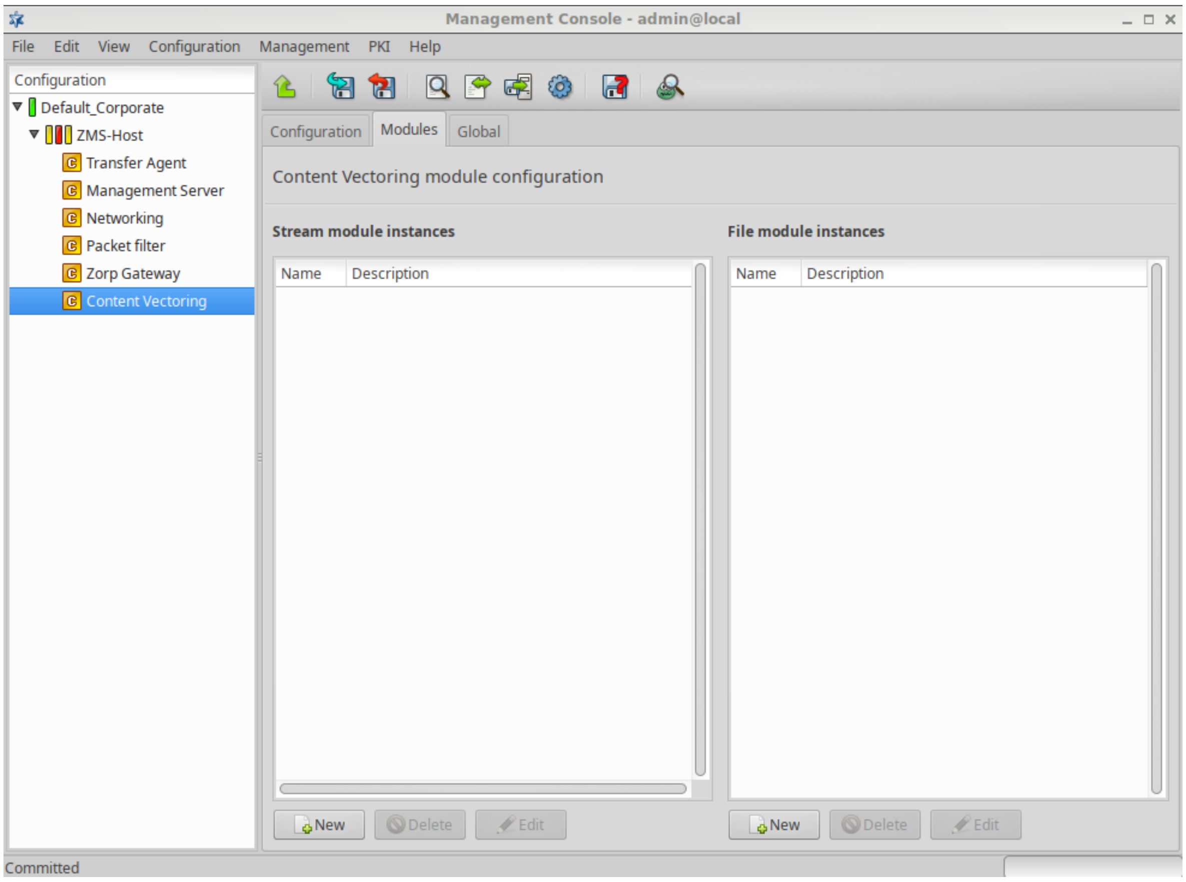


Figure 14.1. The Modules tab of the Content Vectoring component

The panel has two distinct sections, the left window manages stream module instances, the right window file module instances. The functionality and the handling of the two management interface are identical. Existing module instances are shown in the main section of the panel, organized into a tree based on the module they represent (NOD32, html, and so on). Below this list are buttons to create, delete and edit module instances.

14.2.1.1. Procedure – Creating a new module instance

Step 1. Click on the **New** button below the stream and file module instances lists.

Step 2. Select the module to be configured from the **Module** combobox in the window that pops up.

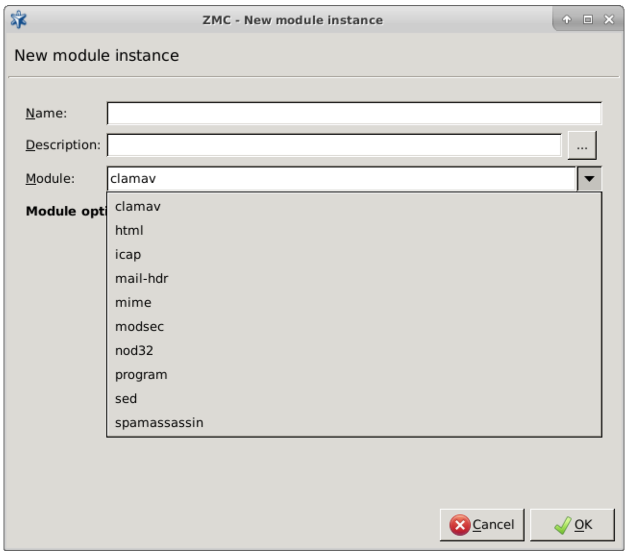
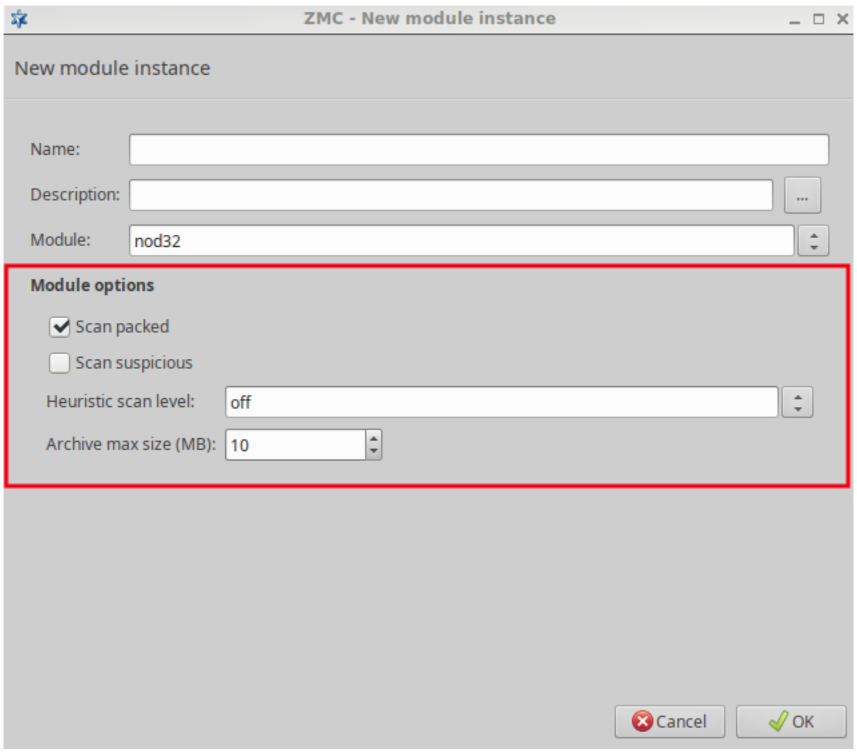


Figure 14.2. Selecting the module

Step 3. Enter a name (and optionally a description) describing the instance.

Step 4. Set the parameters of the module using the **Module options** section of the dialog window. The available options are module-specific and are described in *Section 14.2.1.2, ZCV modules (p. 355)*.



Name:

Description:

Module: nod32

Module options

[✓] Scan packed

[ ] Scan suspicious

Heuristic scan level: off

Archive max size (MB): 10

Figure 14.3. Configuring module options

14.2.1.2. ZCV modules

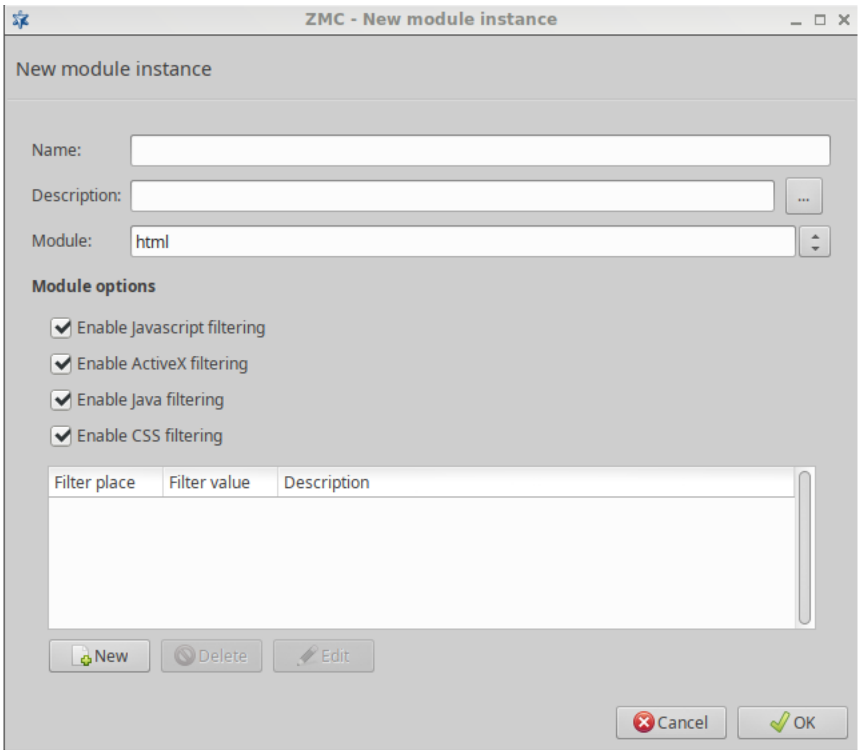
Each module available in ZCV has its own parameters that can be set separately for each module instance. Some of the modules have global options that apply to all instances of that module, these are also described at the particular module.

The clamav module

The `clamav` module uses the Clam AntiVirus engine to examine incoming files. It supports only the file mode and only detects infected files; it does not attempt to disinfect them. Starting with Zorp 3.4, the module automatically scans archived files as well.

The HTML module

The HTML module can be used to filter various scripts and tags in HTML pages. It can operate both in file and stream modes.



Name:

Description:

Module: html

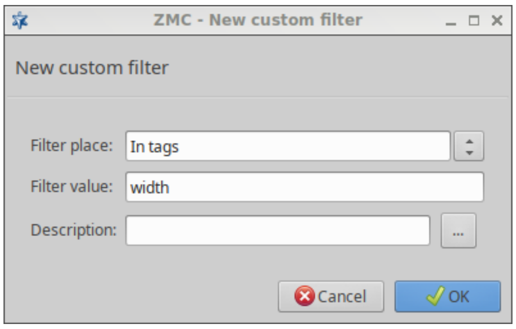
Module options

- [✓] Enable Javascript filtering
- [✓] Enable ActiveX filtering
- [✓] Enable Java filtering
- [✓] Enable CSS filtering

Figure 14.4. The HTML module

The HTML module has the following options:

- **Enable JavaScript filtering**: Remove all JavaScripts. Enabling this option removes all *`javascript`* and *`script`* tags, and the conditional value prefixes (for example, *`onclick`*, *`onreset`*, and so on).
- **Enable ActiveX filtering**: Remove all ActiveX components. Enabling this option removes the *`applet`* tags and the *`classid`* value prefix.
- **Enable Java filtering**: Remove all Java code references. Enabling this option removes the *`java:`* and *`application/java-archive`* inclusions, as well as the *`applet`* tags.
- **Enable CSS filtering**: Remove cascading stylesheet (CSS) elements. Enabling this option removes the single *`link`* tags, the *`style`* tags and options, as well as the *`class`* options.
- **Filter HTML tags**: Custom filters can be added to remove certain elements of the HTML code using the **New** button. The filter can remove the specified values from HTML tags, single tags, options and prefixes (specified through the **Filter place** combobox). The **Filter value** specifies the name of the tag/header to be removed.


ZMC - New custom filter

New custom filter

Filter place: In tags

Filter value: width

Description:

Figure 14.5. Filtering HTML tags

The **Filter place** parameter has the following options:

In tags: Remove everything between the specified tag and its closing tag. Embedded structures are also handled.

In single tags: Remove all occurrences of the specified single tag. A single tag is a tag that does not have a closing element, for example, *`img`*, *`hr`*, and so on.

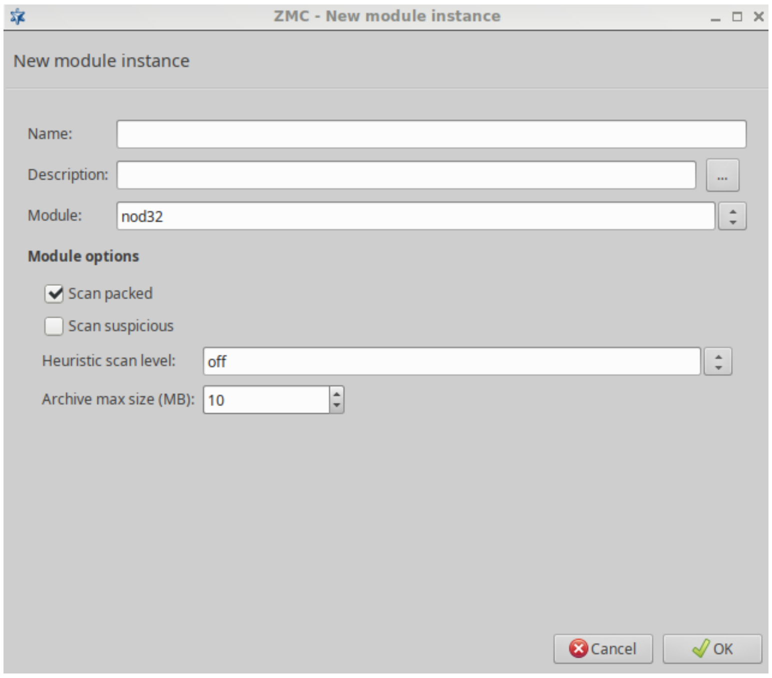
In options: Remove options and their values, for example, *`width`*, and so on.

In prefixes: Remove all options starting with the string set as **Filter value**. The *`on`* option will, for example, remove all options like *`onclick`*, and so on.

Note
The HTML module is designed to process only text data. It cannot handle binary data, thus directing binary files to the module should be avoided.

The NOD32 module

The *`NOD32`* module uses the NOD32 virus filtering engine to examine incoming files. It supports only the file mode.



New module instance

Name:

Description:

Module: nod32

Module options

☑ Scan packed

☐ Scan suspicious

Heuristic scan level: off

Archive max size (MB): 10

Figure 14.6. The NOD32 module

The module has the following parameters:

- **Scan packed**: It enables or disables virus scanning on archived files.
- **Scan suspicious**: It enables or disables virus scanning on suspicious files (for example, suspicious files are often new variants of known viruses).
- **Heuristic scan level**: It defines the level of heuristic (non-database based) sensitivity. The available levels are *`OFF`*, and *`NORMAL`*.
- **Archive max size**: It defines the maximum unpacked size (megabytes) of a single archive scanned. If a 2.5 MB .zip file, for example, contains a file that is 80 MB uncompressed, and the **Archive max size** option is set to 10 MB, the file will not be scanned for viruses. However, if the **Archive max size** option is set to 100 MB, ZCV will scan the file.

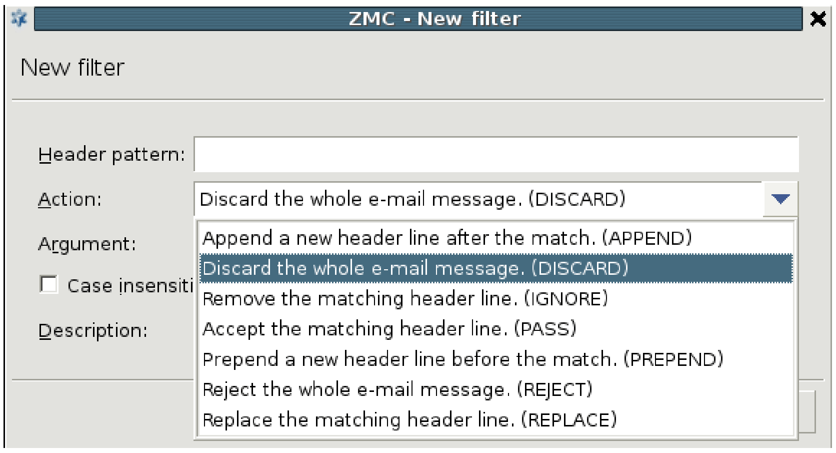
Note
The ESET's NOD32 module tries to resolve reverse host names of all the locally assigned IP addresses, on static, VLAN and TUN interfaces (except for loopbacks) for licensing purposes at ZCV startup. To ensure the fastest ZCV startup and restart, the reverse host names must be available through DNS service or via the *hosts* file. If there is a DNS service but the reverse names are not available, quick *`NXDOMAIN`* responses are sufficient as well. Without a DNS service the NOD32 plugin does not work and gives license activation related error log messages.

The mail header filtering (mail-hdr) module

The *`mail-hdr`* module can filter and maniputale e-mail headers in both stream and file modes. It scans the incoming e-mail (stream or file) using regular expressions and deletes or modifies the matching headers. New headers can also be inserted into the mails.

Warning
E-mail headers are processed and manipulated line-by-line. However, a header can span multiple lines.

A single instance can include multiple filters; the order, these filters are processed, can be set using the arrow buttons. Each filter consists of a pattern, an action that is performed when the pattern is found, and an argument (for example, a replacement header). Note that not every action requires an argument.



Discard the whole e-mail message. (DISCARD)

Append a new header line after the match. (APPEND)
Discard the whole e-mail message. (DISCARD)
Remove the matching header line. (IGNORE)
Accept the matching header line. (PASS)
Prepend a new header line before the match. (PREPEND)
Reject the whole e-mail message. (REJECT)
Replace the matching header line. (REPLACE)

Figure 14.7. Filtering mail headers

A filter has the following parameters:

- **Header pattern**: It is the search string to be found in the headers. Regular expressions can be used. The following options are also available for regular expressions:
- **Action**: It is the action to be performed on the header line or the whole message if the pattern is found in the message. The following actions are available:
  - *Append*: Add the argument of the filter as a new header line after the match.
  - *Discard*: Discard the entire e-mail message. The argument is returned to the mail server sending the message as an error message.
  - *Ignore*: Remove the matching header line from the message.
  - *Pass*: Accept the matching header line. This action can be used to create exceptions from other filter rules.
  - *Prepend*: Add the argument of the filter as a new header line before the match.
  - *Reject*: Reject the entire e-mail message. The argument is returned to the sender of the message as an error message.
  - *Replace*: Replace the matching header line to the argument of the filter.
- **Argument**: The **Regular expression** will be replaced with this string if found in the stream. The replacement can contain \n (n being a number from 1 to 9, inclusive) references, which refer to the portion of the match which is contained between the nth \(and its matching \). Also, the replacement can contain unescaped & characters which will reference the whole matched portion of the pattern space.
- **Case insensitive**: The case sensitive mode can be disabled by selecting this checkbox.

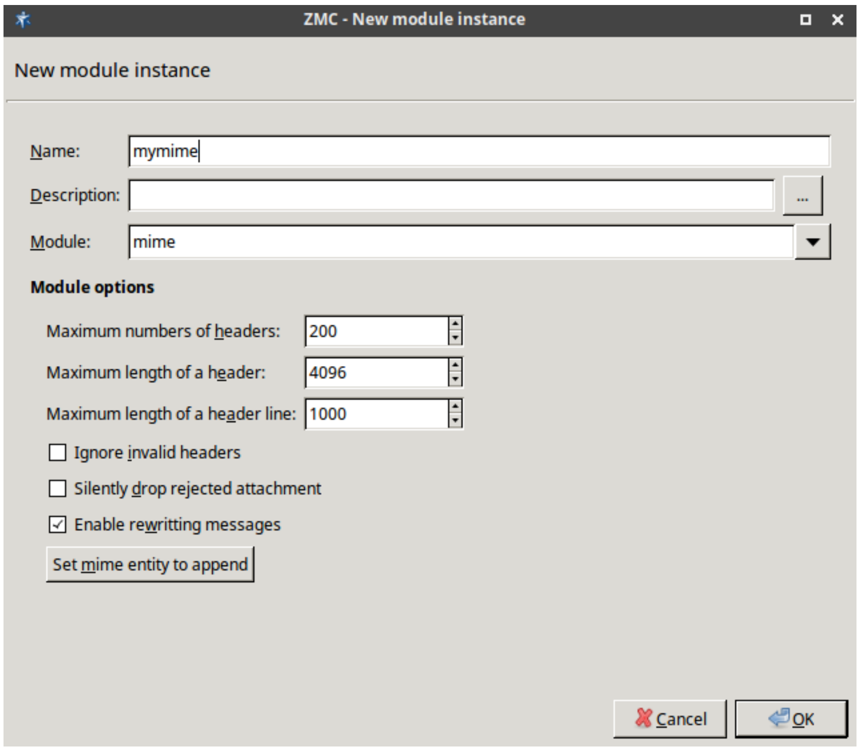
The mime module

The *mime* module inspects and filters MIME objects (that is, mail attachments). It can check the MIME headers that describe the objects for validity, and also call a virus filtering ZCV module to scan the object for viruses. The mime module supports only file mode. The module has the following parameters:

- **Maximum number of headers**: It is the maximal number of headers permitted in a MIME object. The object is removed if it exceeds this limit.
- **Maximum length of a header**: It is the maximal length of a header in characters. It applies to the total length of the header. The header is removed if it exceeds this limit.
- **Maximum length of a header line**: It is the maximal length of a header line in characters. It applies to every single line of the header. The header is removed if it exceeds this limit.
- **Ignore invalid headers**: If it is enabled, headers not complying to the related RFCs or violating the limits set in the previous options are automatically removed (dropped).

Warning
If **Ignore invalid headers** is disabled and an invalid header is found, the entire object (for example, e-mail) is rejected.

- **Silently drop rejected attachment**: By default, the *mime* ZCV module replaces the removed objects (attachments) with the following note that informs the recipient of the message about the removed attachments: *The original content of this attachment was rejected by local policy settings.* If the **Silently drop rejected attachment** option is enabled, no note is added to the e-mail.
- **Enable rewriting messages**: If it is disabled, the *mime* module does not modify the messages.
- **Set mime entity to append**: The *mime* ZCV module can automatically add a MIME object to the inspected messages. To use this feature, verify that the **Enable rewriting messages** option is enabled, select **Set mime entity to append**, paste the MIME object into the appearing dialog box, and select **OK**.


ZMC - New module instance

New module instance

Name: mymime

Description:

Module: mime

Module options

Maximum numbers of headers: 200

Maximum length of a header: 4096

Maximum length of a header line: 1000

☐ Ignore invalid headers

☐ Silently drop rejected attachment

☑ Enable rewritting messages

Figure 14.8. Options of the mime module

To scan the actual MIME objects (for example, the attachments of an e-mail) for viruses, a special rule group has to be created, called *`mime-data`*. Use this as the name of the rule group, and add a virus filtering module (for example, *`clamav`*) to this rule group. When the *`mime`* module is scanning an e-mail message, it inspects the attachments, then pass the attachment to the *`mime-data`* rule group to scan for viruses. See *Section 14.2.3, Routers and rule groups (p. 370)* for details on creating rule groups.

The program module

The *`program`* module is a general wrapper for third-party applications capable of working in stream or file mode. The stream or file is passed to the application set in the **Program** field.

A single instance can include multiple filters; the order these filters are processed can be set using the arrow buttons.

A program module has the following parameters:

- **Program**: It is the application to be executed.
- **Timeout**: If the application set in the **Program** field does not provide a return value within this interval, it is assumed to be frozen.
- **The program may modify the data**: The program may make changes to the data and return the modified version to ZCV.

The stream editor (sed) module

The *sed* module is a stream editor capable of working in both stream and file mode. It scans the target stream and replaces the string to be found (specified as a regular expression) with another string.

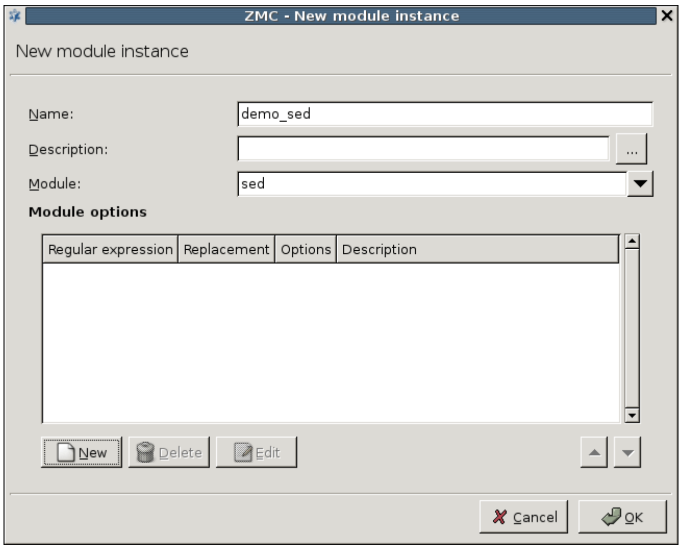


Figure 14.9. The stream editor module

Warning
This module is similar to, but not identical with the common UNIX `sed` command.

A single instance can include multiple filters; the order these filters are processed can be set using the arrow buttons.

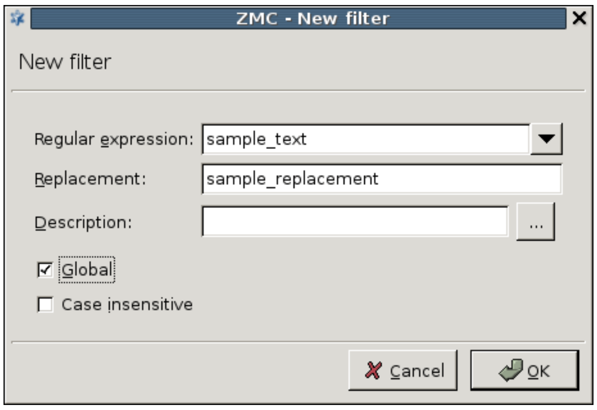


Figure 14.10. Filtering with the stream editor module

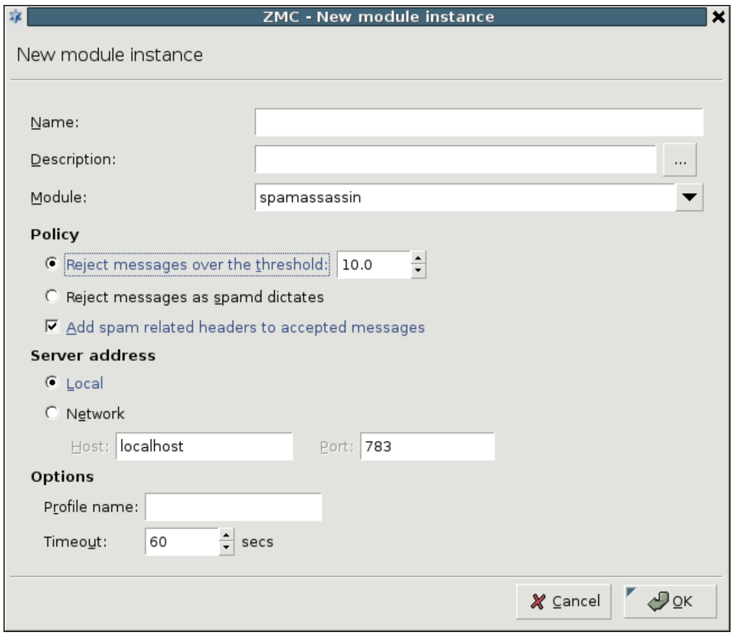
A filter has the following parameters:

- **Regular expression**: It is the search string to be found in the stream. Regular expressions can be used. The following options are also available for regular expressions:

- **Replacement**: The **Regular expression** will be replaced with this string if found in the stream. The replacement can contain \n (n being a number from 1 to 9, inclusive) references, which refer to the portion of the match which is contained between the nth \(and its matching \). Also, the replacement can contain unescaped & characters which will reference the whole matched portion of the pattern space.
- **Global**: Replace all occurrences of the search string. If it is not checked in, the filter will replace only the first occurrence of the string.
- **Case insensitive**: Disable case sensitive mode.

The spamassassin module

The *`spamassassin`* module uses the Spamassassin spam filtering engine to examine incoming e-mails. It supports only the file mode.


ZMC - New module instance

New module instance

Name:

Description:

Module: spamassassin

Policy

(●) Reject messages over the threshold: 10.0

( ) Reject messages as spamd dictates

[✓] Add spam related headers to accepted messages

Server address

(●) Local

( ) Network

Host: localhost Port: 783

Options

Profile name:

Timeout: 60 secs

Figure 14.11. The Spamassassin module

The module has the following parameters:

- Policy-related options
  - *Reject messages over the threshold*: Reject the message only if it is in spam status. By default, SpamAssassin rejects all e-mails with a spam status (called *`required_score`* in SpamAssassin terminology) higher than 5 as spam. However, to minimize the impact of false positive alarms, if the spam status of an email (as calculated by SpamAssassin) is over the *`required_score`*, but below the value set in threshold, ZCV only marks the e-mail as spam, but does not reject it. If the spam status of an e-mail is above the threshold, it is automatically rejected.
  - *Reject messages as spamd dictates*: Reject all e-mails detected as spam by SpamAssassin.
  - *Add spam related headers to accepted messages*: Append headers to the e-mail containig information about SpamAssassin, the spam status of the e-mail, and so on. Sample headers are presented below.

```
X-Spam-Checker-Version: SpamAssassin 3.0.3 (2005-04-27) on
mailserver.example.com X-Spam-Level: X-Spam-Status: No, score=-1.7
required=5.0 tests=BAYES_00 autolearn=ham version=3.0.3
```

- Server address
  - *Local*: SpamAssassin is running on the same host as ZCV. In this case, communication is performed through a UNIX domain socket.
  - *Network*: SpamAssassin is running on a remote machine. Specify its address and the port SpamAssassin is accepting connections in the **Host** and **Port** fields, respectively.
- Other options
  - *Profile name*: The user under which SpamAssassin should filter e-mails. Default value: not set, the user running SpamAssassin is used (usually *nobody*).
  - *Timeout*: It is the timeout value for SpamAssassin.

The ModSecurity module

Zorp Professional is already capable of providing protection for various web servers with SSL termination, however, the proxy, controlling the HTTP protocol is responsible for following the RFC, the Zorp Content Vectoring System System NOD32 and other modules are responsible for the virus filtering of the transferred content. Now, these solutions can be complemented with a web application-level security gateway module.

ModSecurity can be integrated to Zorp's HTTP proxy with the help of the Zorp Content Vectoring System (ZCV) module. It ensures an additional, independent level of protection for the web servers, achieving this with the help of the transferred HTTP headers, the concurrent analysis of data and the application of the relevant policies (Free: 'OWASP ModSecurity Core Rule Set (CRS) Version 3' or professional: 'Commercial Rules from Trustwave SpiderLabs'). With the help of this solution the malware and non-trustworthy HTTP requests get blocked usually already on the Zorp Professional and cannot reach the web server.

14.2.2. Creating scanpaths

Scanpaths are lists of module instances and some settings (trickling mode, quarantining, and so on) specific to the given scanpath. Traffic directed to a scanpath is inspected by the module instances specified in the scanpath. Scanpaths can include both stream and file modules, but always the stream modules process the data first.

Scanpaths can be managed (created, deleted and edited) from the **Scanpath** section of the **Configuration** tab of the **Content Vectoring** ZMC component. To configure a new scanpath, the following procedure must be completed:

14.2.2.1. Procedure – Creating a new scanpath

Step 1. Click on **New** in the **Scanpath** section of the **Configuration** tab of the **Content Vectoring** ZMC component.

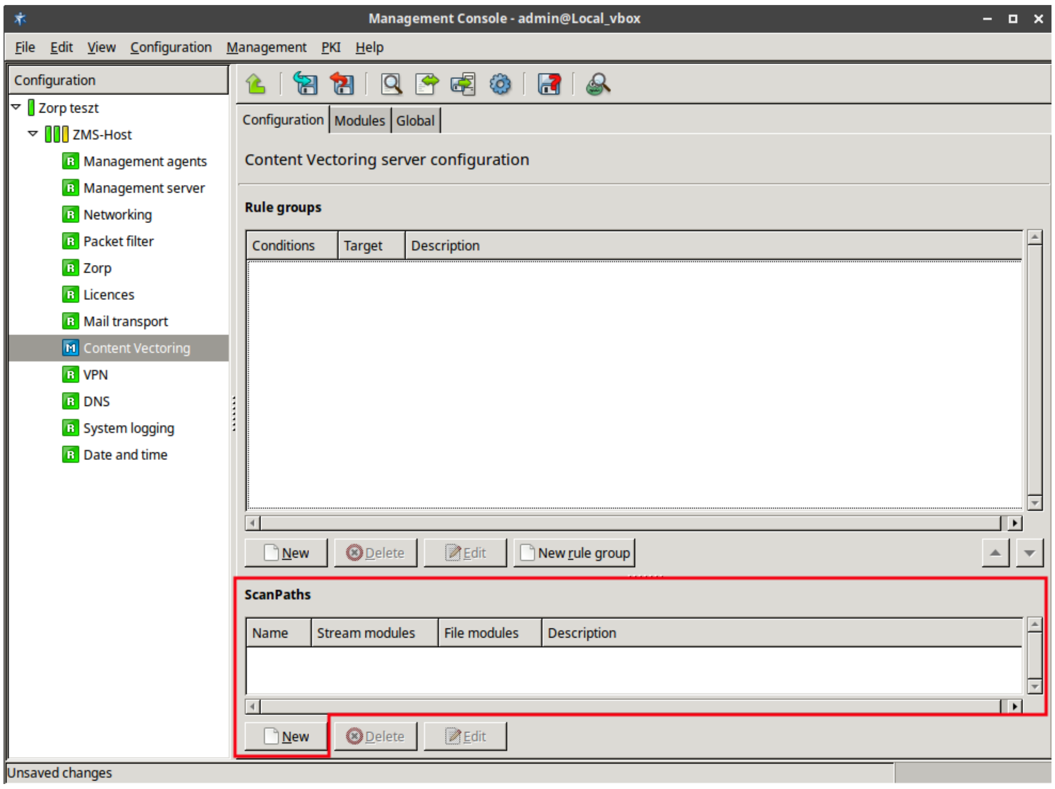
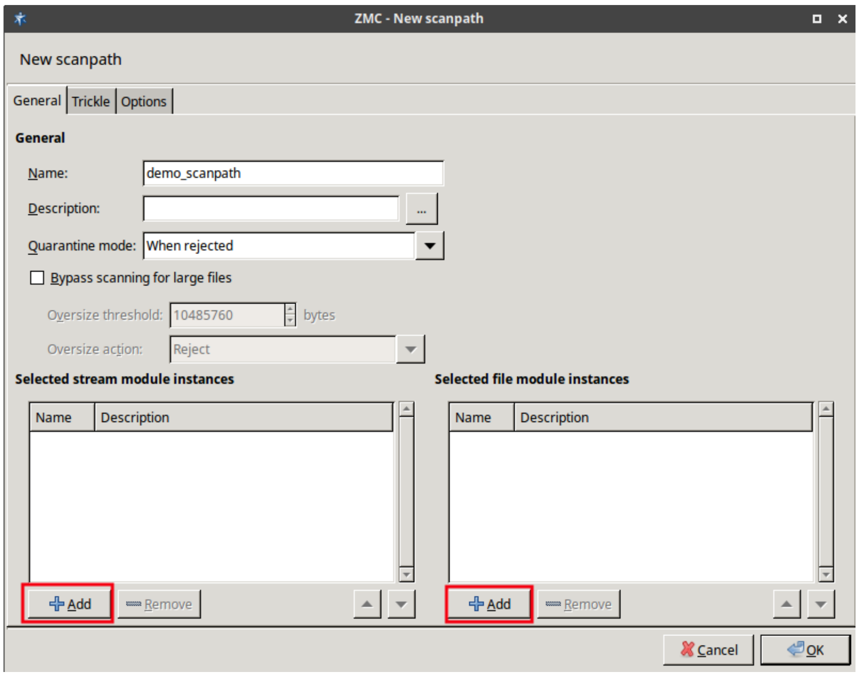


Figure 14.12. Creating a new scanpath

Step 2. Enter a name (and optionally a description) for the scanpath.

Step 3. Use the **Add** buttons and select the stream and/or file module instances to be added to the scanpath.



ZMC - New scanpath

New scanpath

General | Trickle | Options

General

Name: demo_scanpath

Description:

Quarantine mode: When rejected

☐ Bypass scanning for large files

Oversize threshold: 10485760 bytes

Oversize action: Reject

Selected stream module instances

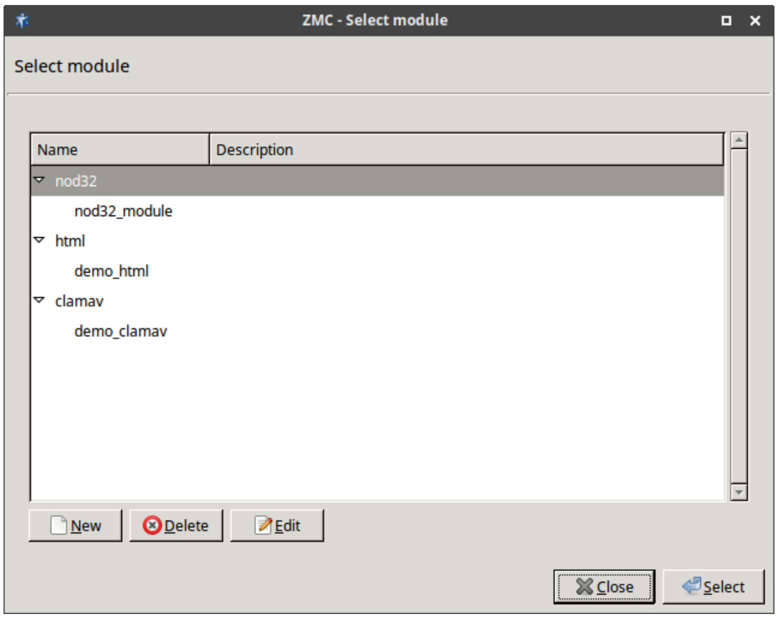
Name	Description

Selected file module instances

Name	Description

Figure 14.13. Adding module instances to a scanpath

It is not necessary to use existing module instances, they can also be created on-the-fly using the **New** button of the **Module instance selection** dialog window.



ZMC - Select module

Select module

Name	Description
nod32	
nod32_module	
html	
demo_html	
clamav	
demo_clamav	

Figure 14.14. Selecting module instances

The data is processed in the order the module instances are listed (starting always with the stream modules). The order of instances can be changed using the arrow buttons below the lists.

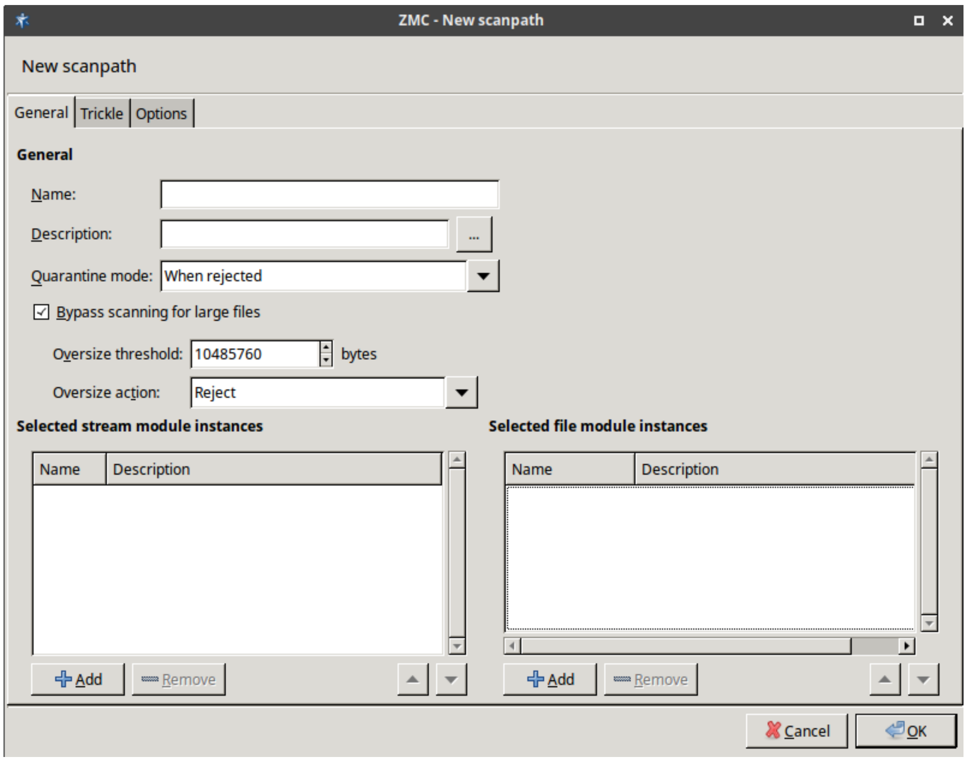
Step 4. Set and configure the quarantine and trickling mode to be used for the scanpath. Also set the policy for large files. These options are described in *Section 14.2.2.2, Scanpath options (p. 367)*.

14.2.2.2. Scanpath options

The following sections describe the options available for scanpaths. The options can be configured on the **General**, **Trickle**, **Options** tabs of the scanpath dialog.

Quarantine and oversized file options

Quarantine mode specifies when an infected object has to be put into quarantine. The original file is always stored.


ZMC - New scanpath

New scanpath

General | Trickle | Options

General

Name:

Description:

Quarantine mode: When rejected

☑ Bypass scanning for large files

Oversize threshold: 10485760 bytes

Oversize action: Reject

Selected stream module instances

Name	Description

Selected file module instances

Name	Description

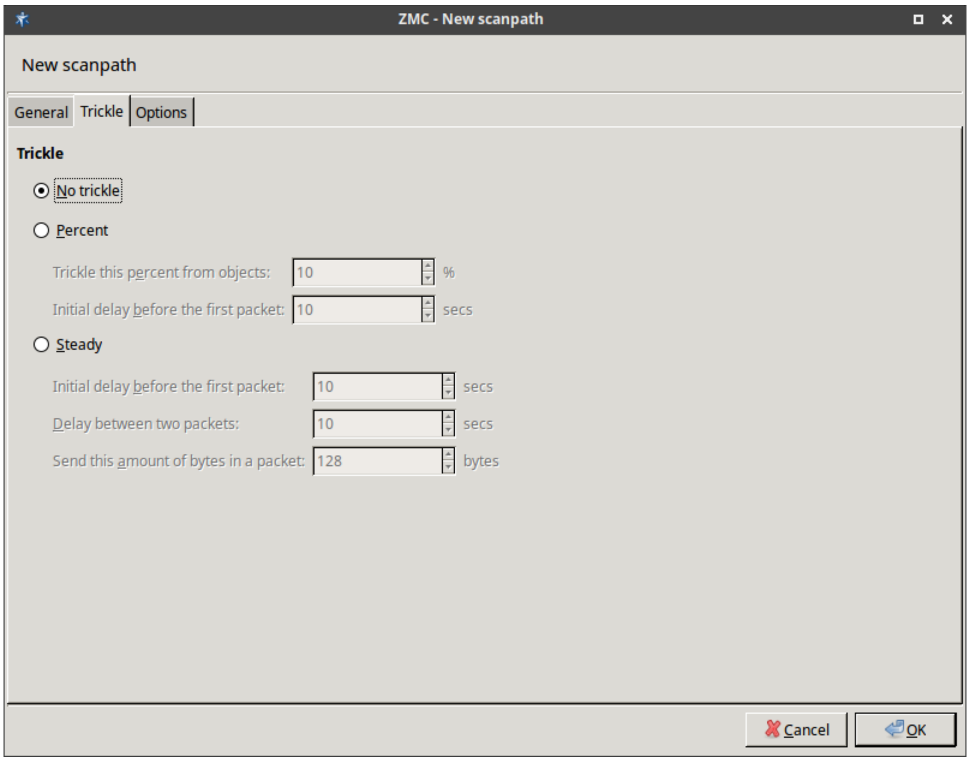
Figure 14.15. Configuring quarantine policy

- **Always**: Quarantine all objects.
- **When rejected**: Quarantine objects that could not be disinfected or rejected for any reasons.
- **When modified or rejected**: Quarantine the modified or infected objects. Modification is completed by 'sed' and 'mailhdr' modules. Quarantine only the original version of the files which have been successfully disinfected. For example, if an infected object is found but it is successfully disinfected, the original (infected) object is quarantined. This way, the object is retained even if the disinfection damages some important pieces of information.
- **Never**: Disable quarantining; objects rejected for any reasons are dropped.

Bypass scanning of large files: By default, all files arriving to the scanpath are scanned. However, this might not be optimal for performance reasons, particularly if large files (for example, ISO images) are often downloaded through the firewall. Therefore, it is possible to specify an **Oversize threshold** value and an **Oversize action**. If the **Bypass scanning of large files** checkbox is selected, objects larger than **Oversize threshold** (in bytes) are not scanned, but **accept**ed or **reject**ed, based on the settings in **Oversize action**. It is also possible to return an **error** message for the oversized files.

Configuring trickle mode

Content filtering cannot be performed on partial files — the entire file has to be available on the firewall. The file will only be sent to the client if no virus was found (or the file was successfully disinfected). Instead of receiving the data in a continuous stream, as when connecting to the server “regularly”, the client does not receive any data for a while, then it “suddenly” starts to flow. This phenomena is not a problem for small files, since these are transmitted and checked fast, probably without the user ever noticing the delay, but can be an issue for larger files when the client application might time out. It can also be inconvenient when the bandwidth of the network on the client and server side of the firewall is significantly different. In order to avoid time outs, a solution called *trickling* is used. This means that the firewall starts to send small pieces of data to the client so it feels that it is receiving something and does not time out. For further information on trickling, see the *Virus filtering and HTTP Technical White Paper* available at the Balasys Documentation Page at https://docs.balasys.hu/. ZCV supports the following trickling modes. These can be set on the **Trickle** tab of the scanpath editor dialog.



[✓ No trickle] Percent Steady

Trickle this percent from objects: 10 %
Initial delay before the first packet: 10 secs
Initial delay before the first packet: 10 secs
Delay between two packets: 10 secs
Send this amount of bytes in a packet: 128 bytes

Figure 14.16. Configuring trickling

- **No trickling**: Trickling is completely disabled. This may result in many connection timeouts if the processing is slow, or large files are downloaded on a slow network.

- **Percent**: Determine the amount of data to be trickled based on the size of the object. Data is sent to the client only when ZCV receives new data; the size of the data trickled is the set percentage of the total data received so far.
- **Steady**: Trickle fixed amount of data in fixed time intervals. Trickling is started only after the period set in **Initial delay before the first packet**. If the whole file is downloaded and processed within this interval, no trickling is used.

Tip
It is recommended to use the percent-based trickling method, because the chance of an operable virus trickling through the system unnoticed is higher when steady trickling is used.

Automatic decompression and error handling

To enable MIME-type checking for the files, set the **Force the mime-type detection** checkbox to active state.

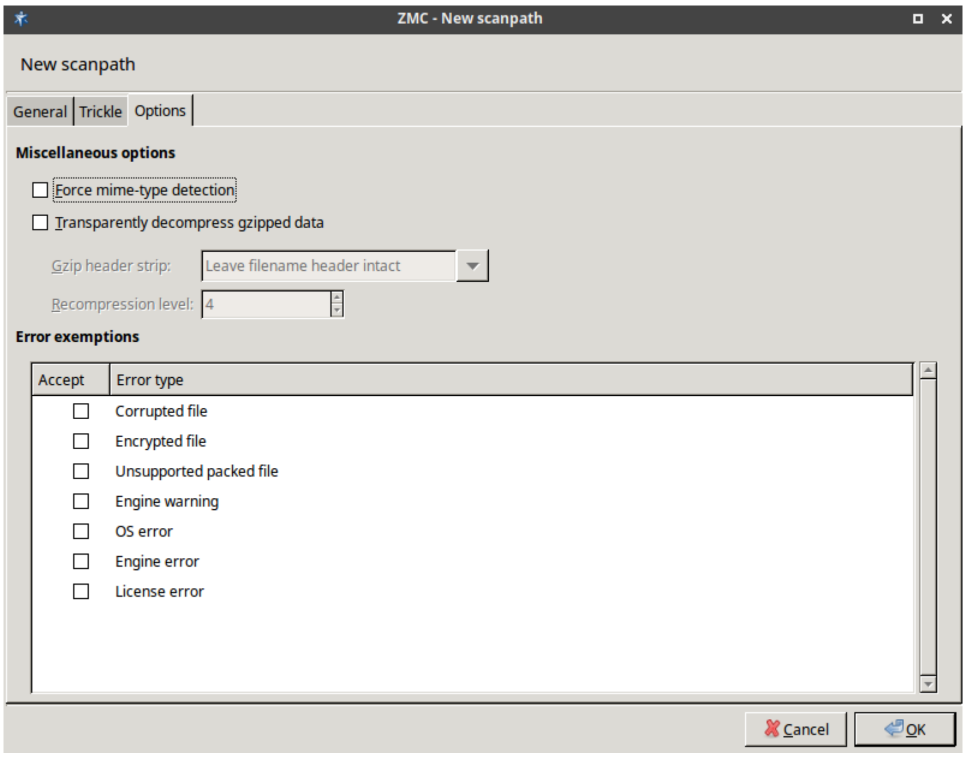
ZCV can automatically decompress gzipped files and transmission and pass the uncompressed data to the modules. After the modules process the data, ZCV can recompress it and return it to Zorp. To enable the automatic decompressing, select the **Transparently decompress gzipped data** checkbox.

The following actions can be set for the gzip headers through the **Gzip header strip** combobox:

- *Leave all gzip headers intact*: Do not modify the headers.
- *Leave filename headers intact*: Retain the headers containing filenames, remove all the other ones.
- *Remove all gzip headers*: Strip all headers.

The compression level of the recompressed data can be set using the **Recompression level** spinbutton.

Note
Compression level of 5 or higher can significantly increase the load on the CPU.


ZMC - New scanpath

New scanpath

General | Trickle | Options

Miscellaneous options

☐ Force mime-type detection

☐ Transparently decompress gzipped data

Gzip header strip: Leave filename header intact

Recompression level: 4

Error exemptions

Accept	Error type
☐	Corrupted file
☐	Encrypted file
☐	Unsupported packed file
☐	Engine warning
☐	OS error
☐	Engine error
☐	License error

Cancel OK

Figure 14.17. Automatic decompression, error handling and mime-type detection

In some cases it is possible that a module or ZCV cannot check an object for some reason (for example the file is corrupted, or the license of the module has expired). In such situations ZCV rejects all objects by default. Exemptions can be set for the errors described below: check the errors for which all objects should be accepted.

- *Corrupted file*: The file is corrupt and cannot be decompressed. Certain virus scanning modules handle encrypted or password-protected files as corrupted files.
- *Encrypted file*: The file is encrypted or password-protected and cannot be decompressed.
- *Unsupported packed file*: The file is compressed with an unknown/unsupported compression method and cannot be decompressed.
- *Engine warning*: The file is suspicious, heuristic virus scanning detected the file as possibly infected.
- *OS error*: A low level error occurred (the module ran out of memory, or could not access the file for some reason).
- *Engine error*: An internal error occurred in the module while scanning the file.
- *License error*: The license of the module has expired.

14.2.3. Routers and rule groups

Routers are simple conditional rules (that is, if-then expressions) that determine how the received object has to be inspected. They consist of a condition and a corresponding action: if the parameter of the traffic (or file) matches the set condition, then the action is performed. The condition consists of a variable and a pattern: the condition is true if the variable of the inspected object is equal to the specified pattern. The action can be a default action (for example, ACCEPT, REJECT, and so on) or a scanpath. Routers cannot be used on their own, they must belong to a rule group. A rule group is a list of routers, defining a set of conditions that are evaluated one-by-one for a given scenario. Rule groups also have a default action or scanpath that is performed if none of the set conditions match the received object. Rule groups are also important because a Zorp proxy can send data only to a rule group, and not to a specific router (see *Section 14.2.4, Configuring Zorp proxies to use ZCV (p. 375)* for details).

Warning
Only the action or scanpath, corresponding to the first matching condition is performed, therefore the order of the routers in a rule group is very important.

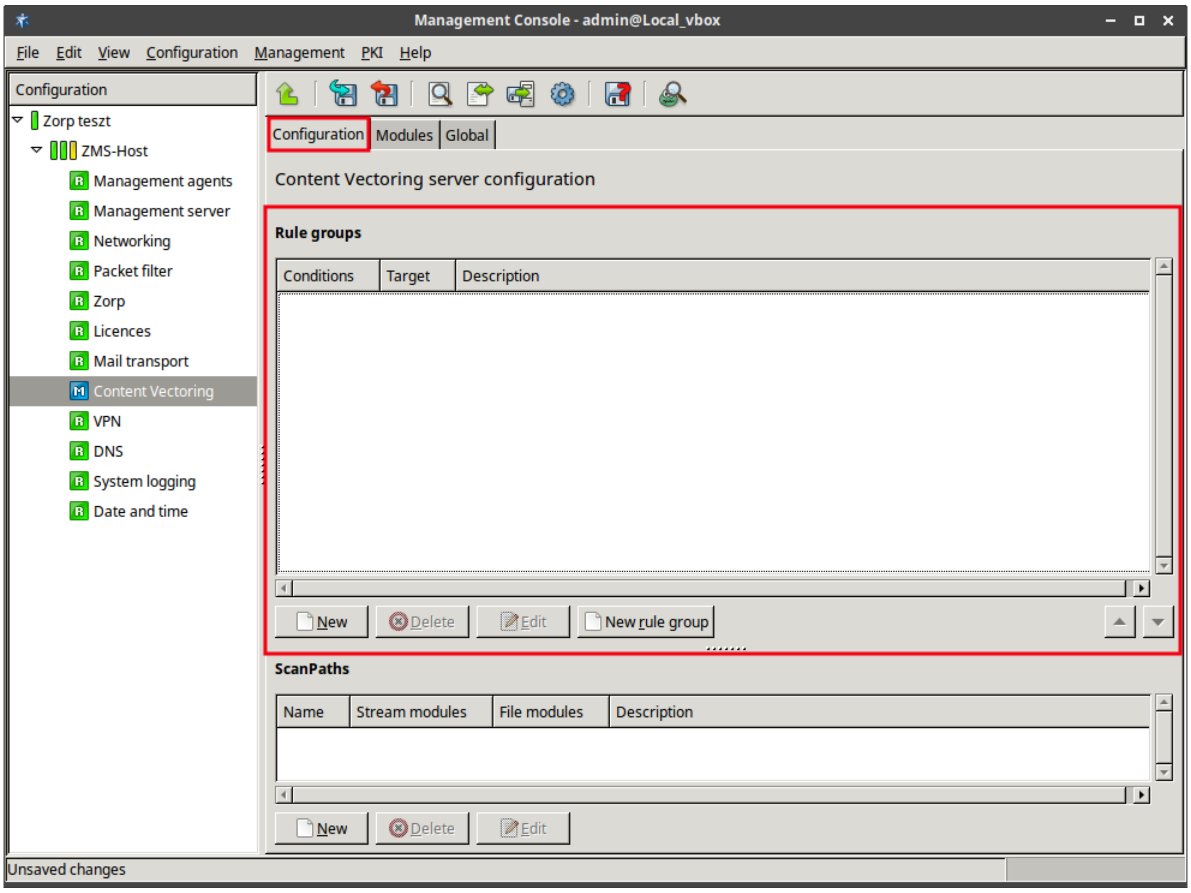


Figure 14.18. Routers and rulegroups

Routers and rule groups can be managed (created, deleted and edited) from the **Rule groups** section of the **Configuration** tab of the **Content Vectoring** ZMC component. The defined rule groups and their corresponding routers (conditions and actions) are displayed as a sortable tree.

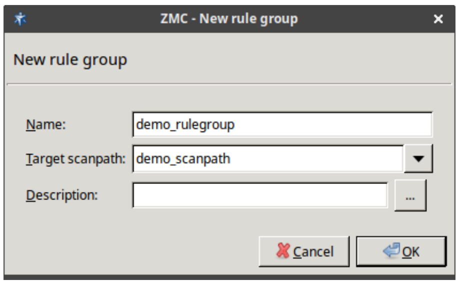
Tip
Rule groups and routers can be disabled from the local menu if they are temporarily not needed.

To create and configure a set of routers, complete the following procedure:

14.2.3.1. Procedure – Creating and configuring routers

Step 1. Navigate to the **Configuration** tab of the **Content Vectoring** ZMC component.

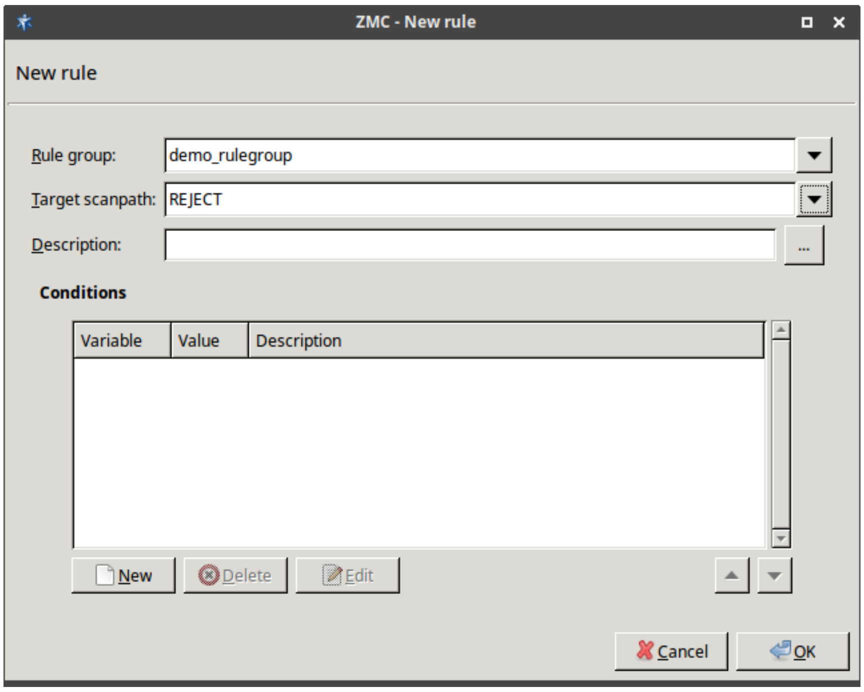
Step 2. Click on **New rule group**. Enter a name for the rule group (scenario) and select a default action or specify a scanpath to be used as default.



Name: demo_rulegroup
Target scanpath: demo_scanpath
Description:

Figure 14.19. Creating a new rulegroup

Step 3. Select the newly created rule group, and click on **New** to add a new router to the group.



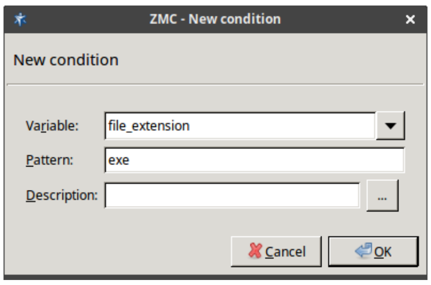
Rule group: demo_rulegroup
Target scanpath: REJECT
Description:

Figure 14.20. Add a new router to the group

Step 4. Select the action to be performed if the conditions match using the **Target scanpath** combobox. The available actions are described in *Section 14.2.3.2, Router actions and conditions (p. 374)*.

Step 5. Click on **New**, and define a condition for the router. Select the variable to be used from the **Variable** combobox, and enter the search term to the **Pattern** field. Wildcards (for example, '*', '?') can be used in the pattern. If the **Variable** of the inspected object matches **Pattern**, the action specified in **Target scanpath** will be performed.

Note
A router can contain multiple conditions. In this case the target action is performed only if all specified conditions are true.


ZMC - New condition

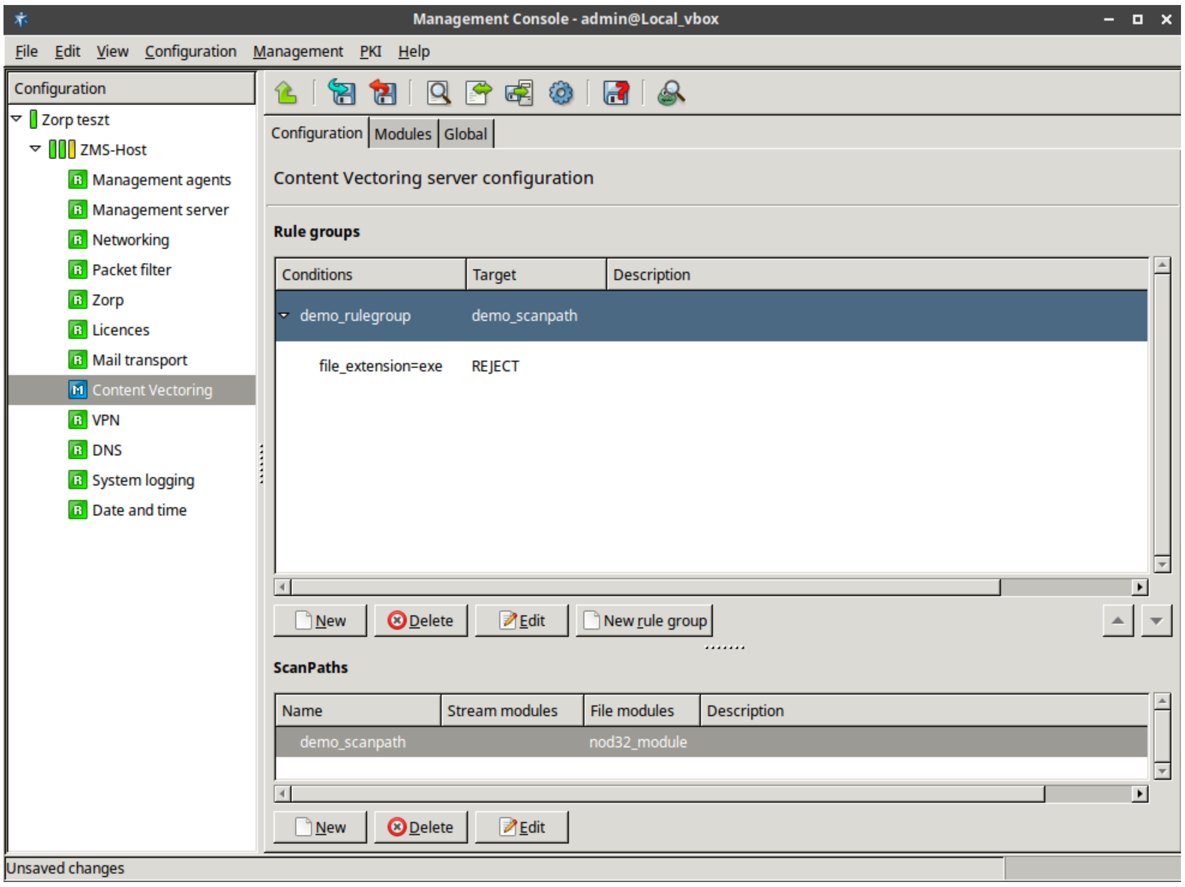
New condition

Variable: file_extension
Pattern: exe
Description:

Figure 14.21. Creating a condition for the router

Step 6. Add as many routers to the rule group as required.

Warning
The routers are evaluated sequentially (much like packet filtering rules), therefore it is important to list them in a correct order. The order of the routers in a rule group can be modified using the arrow buttons below the routers tree. The object is only inspected with the scanpath of the first matching router.



Content Vectoring server configuration

Rule groups

Conditions	Target	Description
demo_rulegroup	demo_scanpath	
file_extension=exe	REJECT	

ScanPaths

Name	Stream modules	File modules	Description
demo_scanpath		nod32_module	

Figure 14.22. A configured scanpath with routers and conditions

14.2.3.2. Router actions and conditions

The following actions are available:

- **ACCEPT**: Accept (allow it to pass the firewall) the object.
- **ACCEPT-QUARANTINE**: Accept the object, but also store a copy of it in the quarantine.
- **REJECT**: Drop the object.
- **REJECT-QUARANTINE**: Drop the object, but store a copy of it in the quarantine. That way, it can be retrieved later if needed.
- **Scanpath**: Inspect the object according to the specified scanpath.

The following table describes some of the variables that can be used in the conditions. This table does not list all such variables, as new variables are added periodically. For an up-to-date list of the available variables see zcv.cfg(5) in *Zorp Professional 7 Reference Guide*, or issue the `man zcv.cfg` command from a shell on the ZCV host.

- *zorp_protocol*: It is the protocol used to transfer the object.
- *file_name*: It is the file name or URL of the object.
- *content_type*: It is the MIME type of the object as specified by the peer.

- *zorp_server_address.ip*: It is the IP address of the server.

Tip
The **Variable** combobox used to create new conditions lists all available variables.

14.2.4. Configuring Zorp proxies to use ZCV

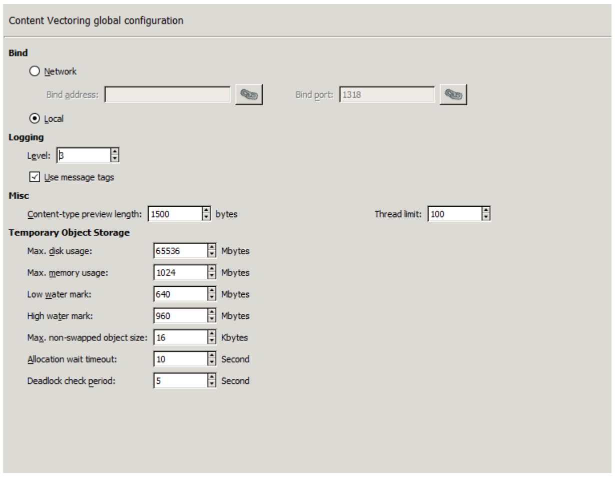
ZCV can only inspect files or streams it receives from Zorp proxies. Zorp proxies send data to ZCV as they would stack another proxy.

The following procedure describes how to configure the communication between Zorp proxies and ZCV.

14.2.4.1. Procedure – Configuring communication between Zorp proxies and ZCV

Step 1. First, the connection settings of ZCV have to be configured in the **Bind** section on the **Global** tab of the **Content vectoring** ZMC component. Specify either the IP address/port pair on which ZCV should accept connections, or the **Local** radiobutton if ZCV will communicate with Zorp through UNIX domain sockets.

Note
The same bind settings will have to be used when the Stacking provider is configured in the **Policies** tab of **Zorp** ZMC component (see *Section 6.7.7, Stacking providers (p. 184)* for details). These settings are required because Zorp and ZCV do not necessarily run on the same hosts.



Content Vectoring global configuration

Bind

[Network ✓ Local]

Bind address:
Bind port: 1318

Logging

Level: 3

✓ Use message tags

Misc

Content-type preview length: 1500 bytes
Thread limit: 100

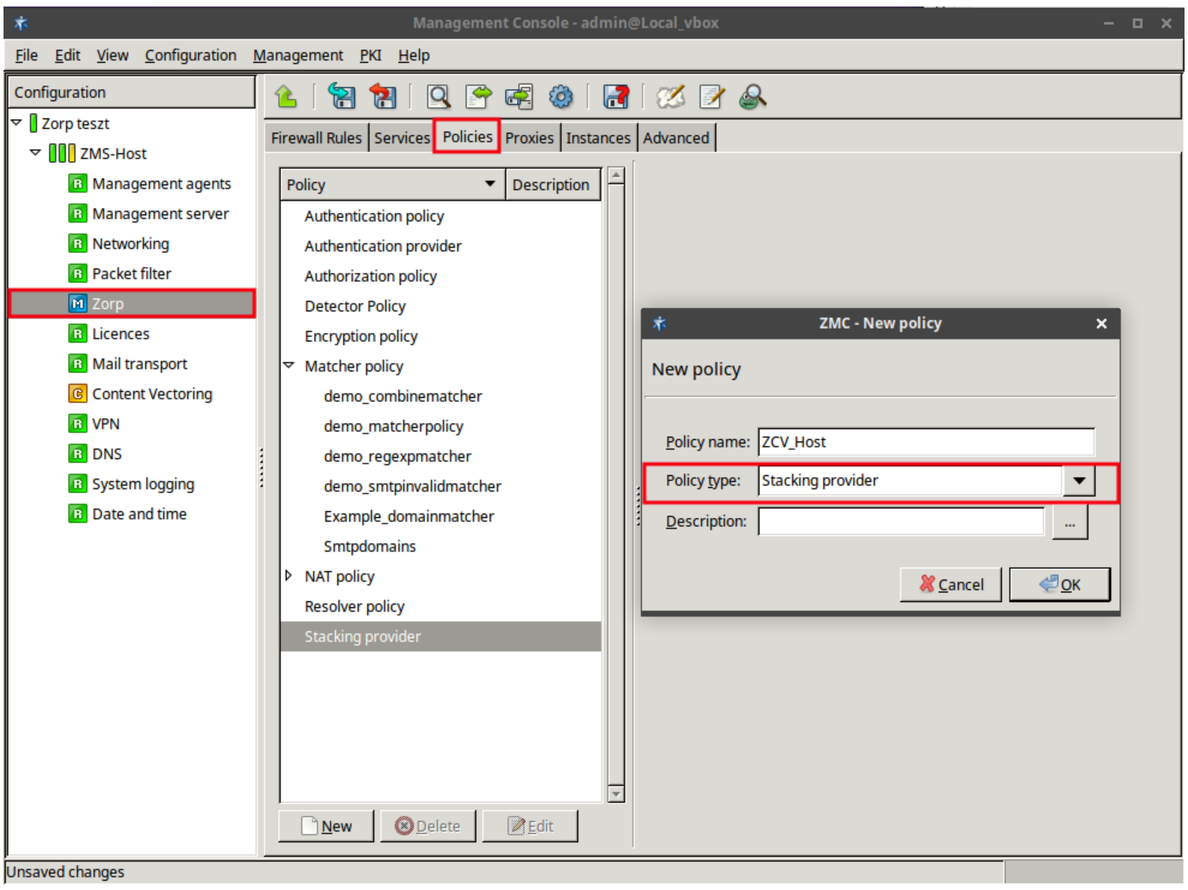
Temporary Object Storage

Max. disk usage:	65536	Mbytes
Max. memory usage:	1024	Mbytes
Low water mark:	640	Mbytes
High water mark:	960	Mbytes
Max. non-swapped object size:	16	Kbytes
Allocation wait timeout:	10	Second
Deadlock check period:	5	Second

Figure 14.23. The connection settings of ZCV

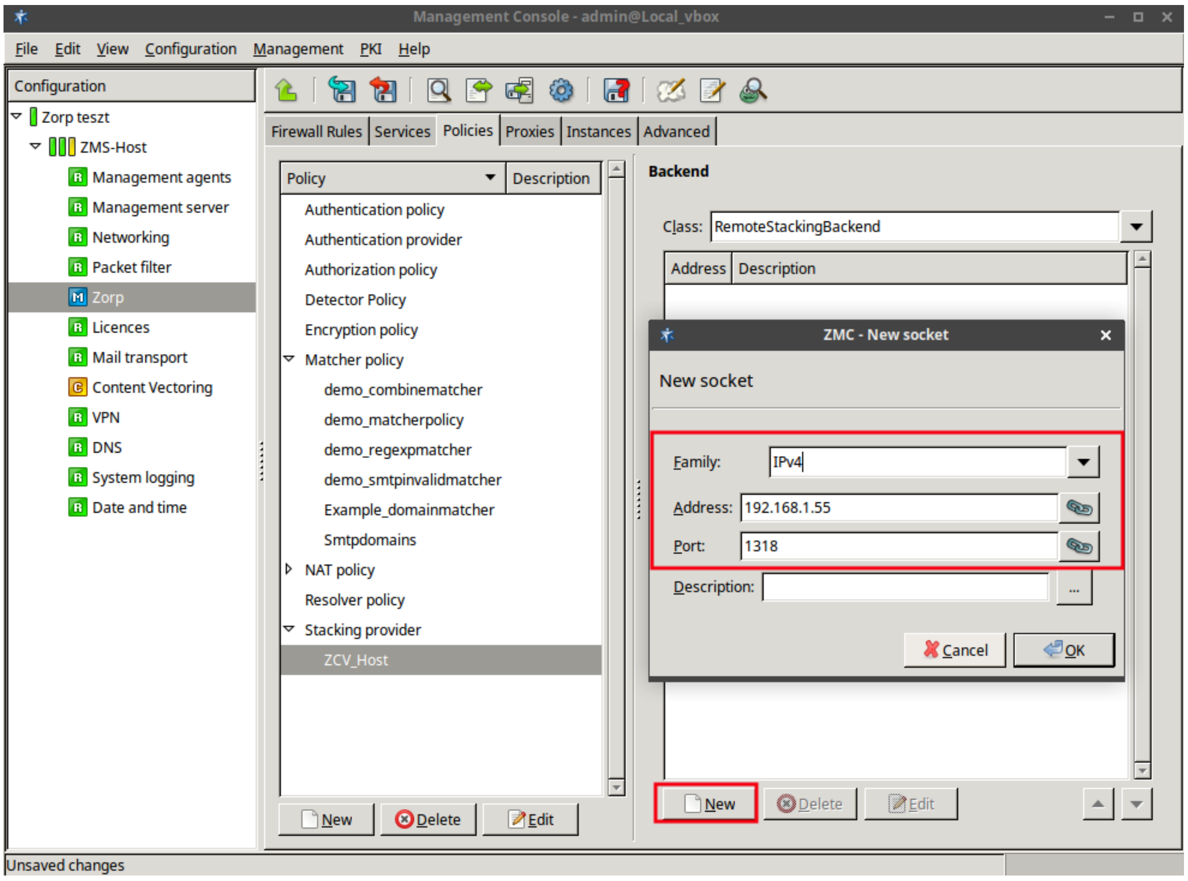
Step 2. Navigate to the **Policies** tab of the **Zorp** ZMC component and create a new *Stacking Provider*. Specify the same connection settings to this stacking provider as set to ZCV in the previous step.

Note
A Stacking provider can contain the connection parameters (that is, IP/port pair) of multiple ZCV hosts. If more than one hosts are specified, Zorp will automatically balance the load sent to these hosts using the round-robin algorithm.



Policy name: ZCV_Host
Policy type: Stacking provider
Description:

Figure 14.24. The connection settings of Zorp and ZCV 1/2

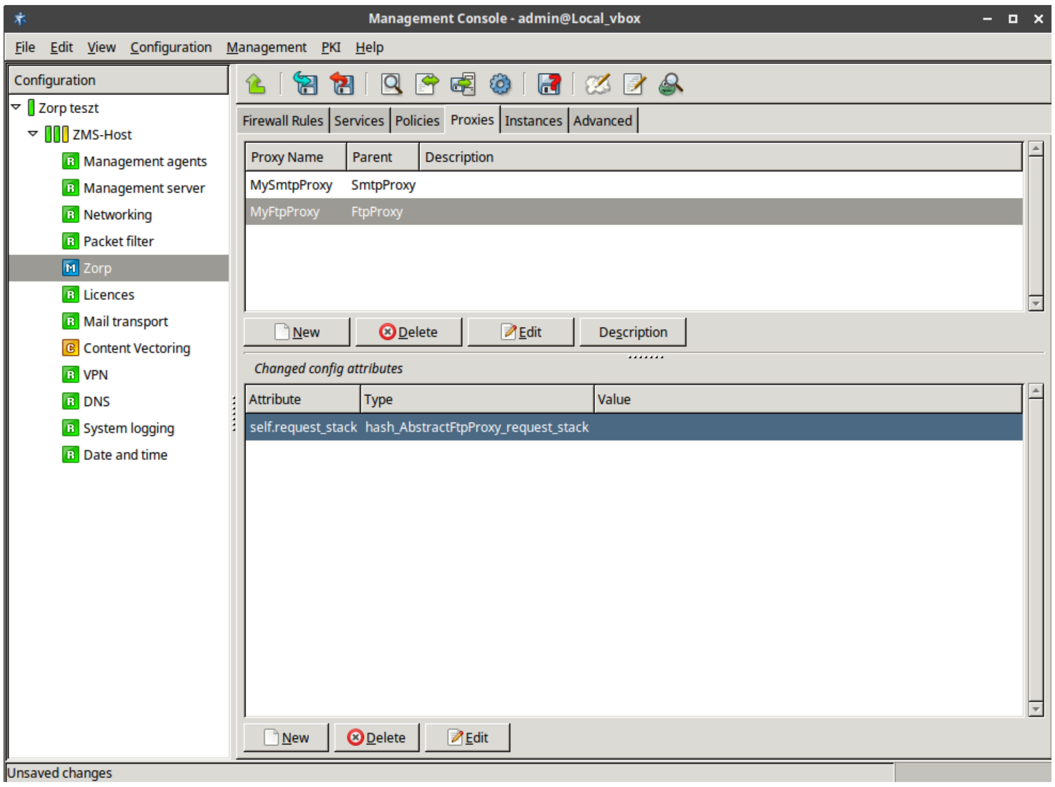


Class: RemoteStackingBackend

Family: IPv4
Address: 192.168.1.55
Port: 1318
Description:

Figure 14.25. The connection settings of Zorp and ZCV 2/2

Step 3. Navigate to the **Proxies** tab of the **Zorp** ZMC component, and select the proxy class that will send the data to ZCV for inspection. This can be an existing or a newly derived proxy class (for example, `MyFtpProxy`).


Management Console - admin@Local_vbox
File Edit View Configuration Management PKI Help
Configuration
Zorp teszt
ZMS-Host
Management agents
Management server
Networking
Packet filter
Zorp
Licences
Mail transport
Content Vectoring
VPN
DNS
System logging
Date and time
Firewall Rules Services Policies Proxies Instances Advanced
New Delete Edit Description
Changed config attributes
New Delete Edit
Unsaved changes

Proxy Name	Parent	Description
MySmtpProxy	SmtpProxy	
MyFtpProxy	FtpProxy	

Attribute	Type	Value
self.request_stack	hash_AbstractFtpProxy_request_stack	

Figure 14.26. Using the Stacking provider in a proxy

Step 4. Add the desired stack attribute of the proxy to the **Changed config attributes** (for example, `self.request_stack`). For details on the stack attributes of the different proxy classes see the description of the proxy class in Chapter 4, Proxies in *Zorp Professional 7 Reference Guide*.

Step 5. Select the stack attribute and click on **Edit**. Click on **New**, and add a key identifying the element of the particular protocol that should be sent over to ZCV for inspection (for example, the `*` parameter). For details, see the description of the proxy class in Chapter 4, Proxies in *Zorp Professional 7 Reference Guide*.

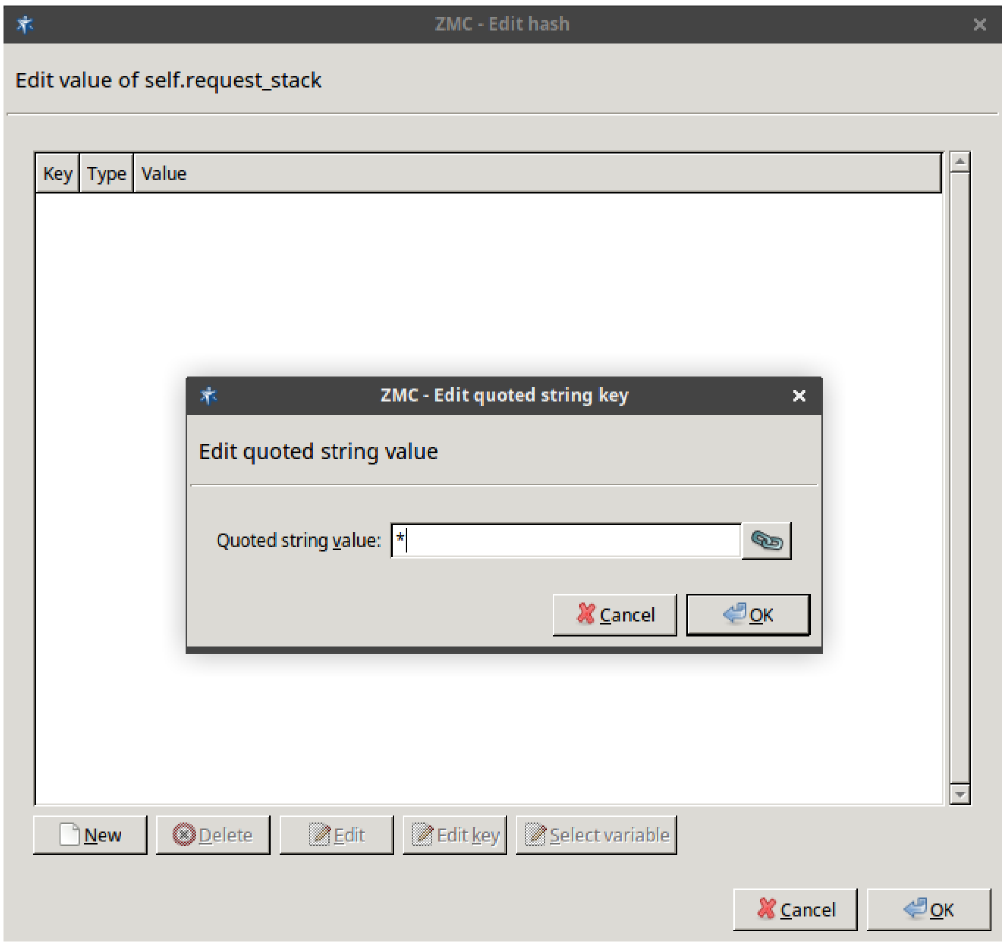
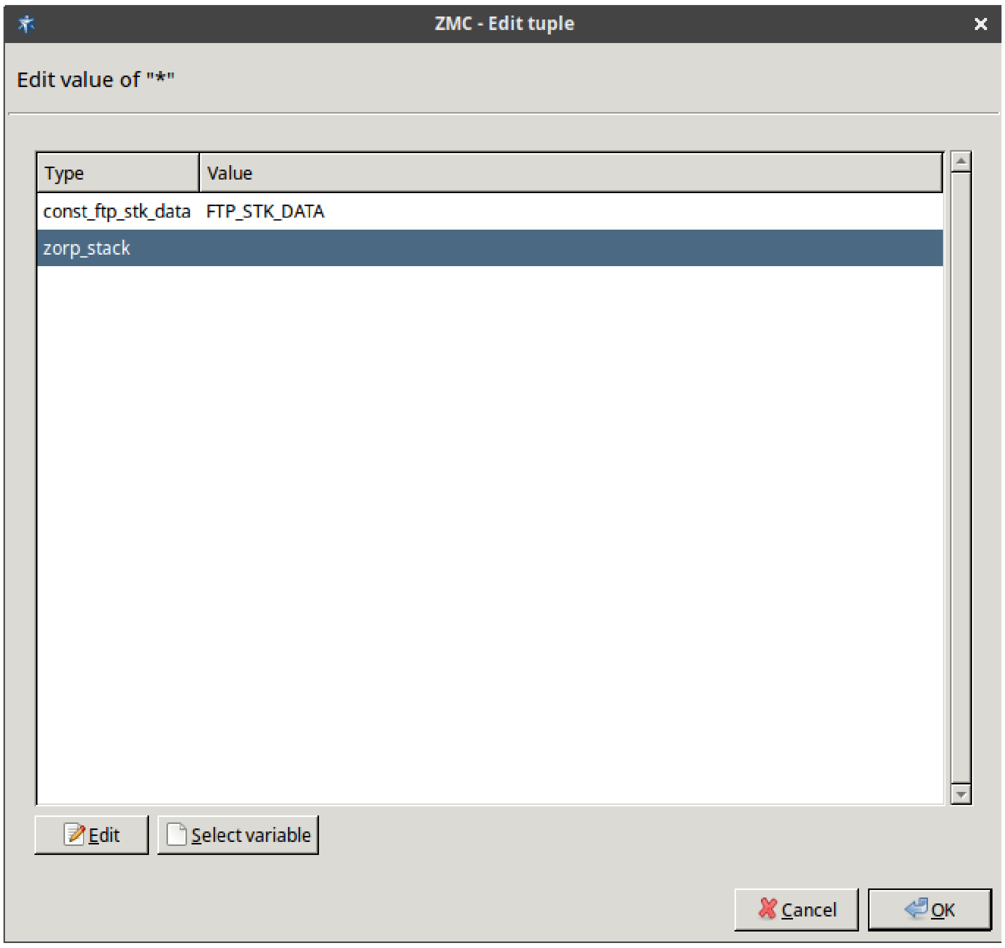


Figure 14.27. Adding a key, identifying an element of a protocol

Step 6. Enable stacking by setting the **Type** attribute to *`type_ftp_stk_data`* of the key using the combobox of the **Type** column, then click **Edit**.

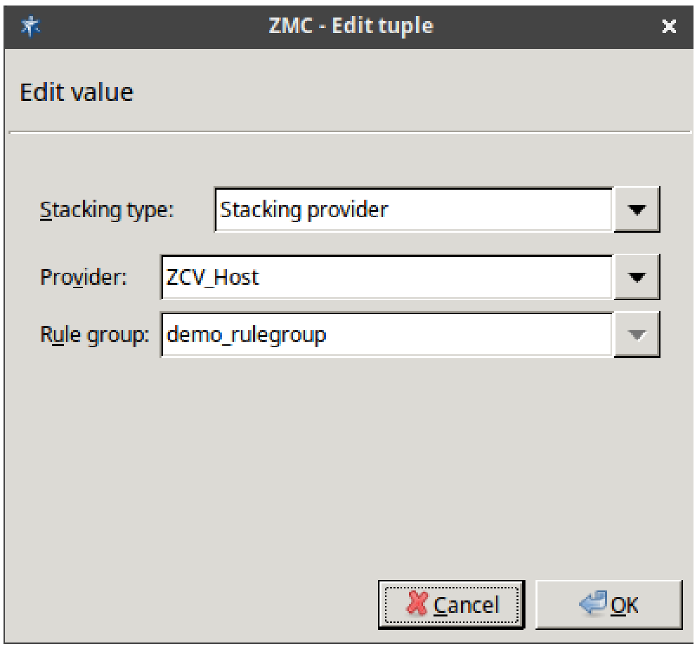
Step 7. Click on **Edit**, select the *`zorp_stack`* attribute in the appearing window, and click again on **Edit**.



Type	Value
const_ftp_stk_data	FTP_STK_DATA
zorp_stack	

Figure 14.28. Stacking a provider

Step 8. Set **Stacking type** to Stacking provider. Select the stacking provider configured in Step 2 from the **Provider** combobox, and the rule group to be used from the **Stacking information** combobox.



Stacking type: Stacking provider
Provider: ZCV_Host
Rule group: demo_rulegroup

Figure 14.29. Selecting the stacking provider and the rulegroup

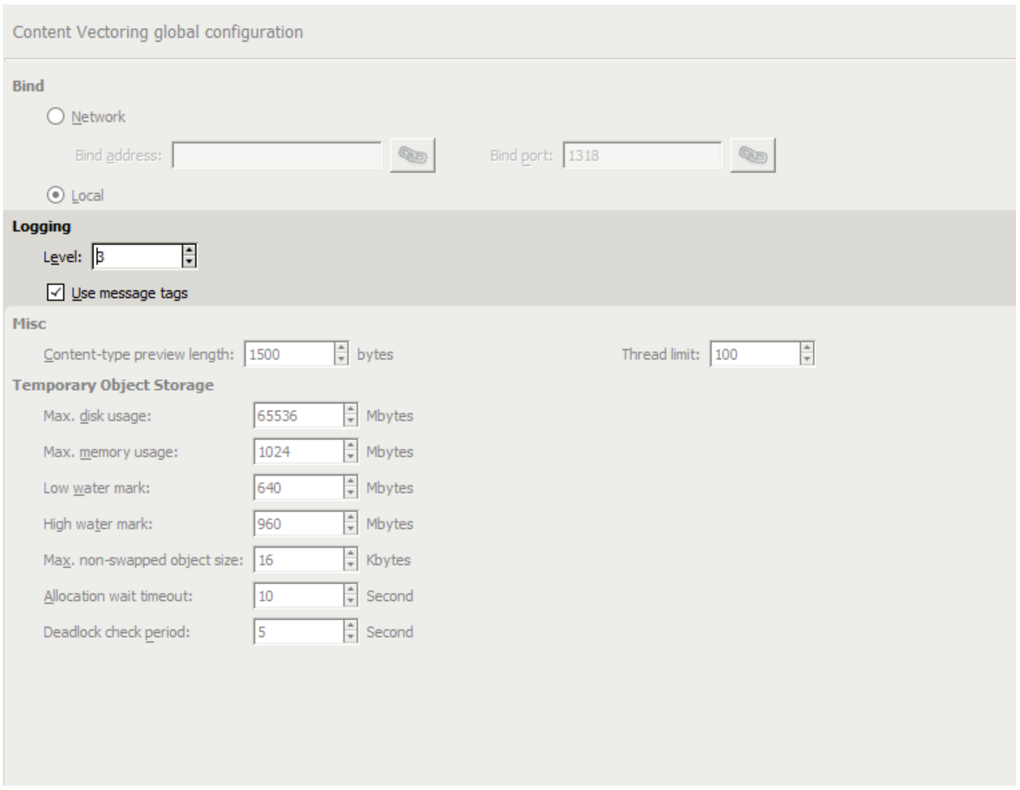
14.2.5. Managing ZCV performance and resource use

A number of global settings affecting the performance and resource use of ZCV can be configured on the **Global** tab of the **Content vectoring** ZMC component. These are discussed in the following sections.

14.2.5.1. Logging in ZCV

The parameters related to logging in ZCV are the following:

- **Log level**: It is the verbosity level of the logs. Level 3 is the default value, and is usually sufficient. Level 0 does not produce log messages, while level 10 logs every small event, and shall only be used for debugging purposes.
- **Use message tags**: Enable the logging of message tags.



Content Vectoring global configuration

Bind

( ) Network

Bind address: Bind port: 1318

(●) Local

Logging

Level: 3

[✓] Use message tags

Misc

Content-type preview length:	1500	bytes	Thread limit:	100	

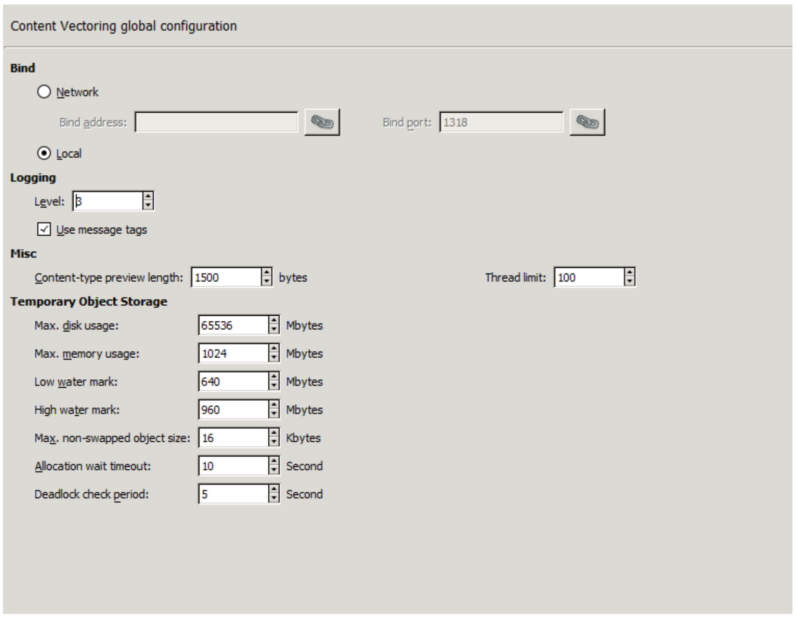
Temporary Object Storage

Max. disk usage:	65536	Mbytes
Max. memory usage:	1024	Mbytes
Low water mark:	640	Mbytes
High water mark:	960	Mbytes
Max. non-swapped object size:	16	Kbytes
Allocation wait timeout:	10	Second
Deadlock check period:	5	Second

Figure 14.30. Configuring the logging of ZCV

14.2.5.2. Memory and disk usage of ZCV

ZCV has a number of options governing the memory and hard disk usage behavior of ZCV. These resources are mainly used to temporarily store the objects while being inspected, decompress archived files, and so on.



Content Vectoring global configuration

Bind

( ) Network

Bind address:

Bind port: 1318

[✓ Local]

Logging

Level: 3

[✓] Use message tags

Misc

Content-type preview length:	1500	bytes
Thread limit:	100	

Temporary Object Storage

Max. disk usage:	65536	Mbytes
Max. memory usage:	1024	Mbytes
Low water mark:	640	Mbytes
High water mark:	960	Mbytes
Max. non-swapped object size:	16	Kbytes
Allocation wait timeout:	10	Second
Deadlock check period:	5	Second

Figure 14.31. Configuring the memory and disk usage in ZCV

The following memory usage settings are available:

- **Max. disk usage**: It defines the maximum amount of hard disk space that ZCV is allowed to use.
- **Max. memory usage**: It sets the maximum amount of memory that ZCV is allowed to use.
- **Low and high water mark**: ZCV tries to store everything in the memory if possible. If the memory usage of ZCV reaches *`high water mark`*, it starts to swap the data onto the hard disk, until the memory usage decreases to *`low water mark`*.
- **Max. non-swapped object size**: Objects smaller than this value are never swapped to hard disk.
- **Content-type preview**: This parameter determines the amount of data (in bytes) read from MIME objects to detect their MIME-type. Higher value increases the precision of MIME-type detection. Trying to detect the MIME-type of objects is required because there is no guarantee that a MIME object is indeed what it claims to be.
- **Thread limit**: This value defines the number of threads ZCV can start. The graphical user interface now sets the ZCV default *Thread limit* to 100. It is possible though to set a different value for the *Thread limit*. Set this value according to the anticipated number of stacked connections. If the *Thread limit* is too low, the Zorp proxies stacking ZCV will experience delays and refused connection attempts. A suggested method to calculate this number, is to monitor the log for "Too many running threads, waiting for one to become free" line and to increase Zorp/ZAS/ZCV *Thread limit* parameter accordingly.
  Note, that if the *Thread limit* is already set with the ZCV "--threads=" option in the init script, then that option takes precedence. The value defined in the init script is cleared with each upgrade though, therefore this value can easily be updated at each upgrade to the preferred value, defined in the GUI.

14.3. Quarantine management in ZMC

All ZCV modules use a common quarantine on each host. The contents of the quarantine on a particular host can be accessed through the **Quarantine** icon that is available on the **Host**, **Zorp**, and **ZCV** components. The main part of this window shows a list of the quarantined files, including columns of meta-information like their date, size, why they were quarantined, and so on. For a detailed list of the possible meta-information see *Section 14.3.1, Information stored about quarantined objects (p. 385)*. The objects in the quarantine can be sorted by clicking on any of these columns. The order of the columns can be simply modified by dragging the column header to its desired place.

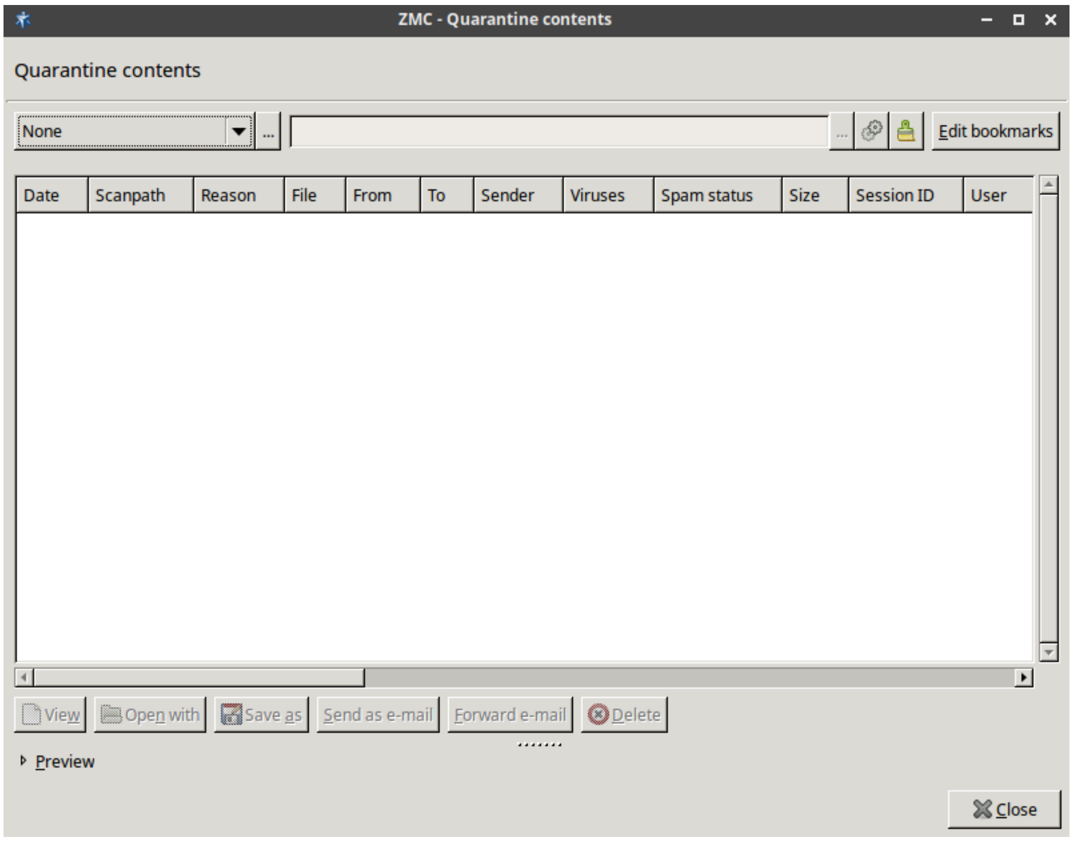


Figure 14.32. The quarantine viewer

The **Quarantine contents** window is a *Filter window*, thus various simple and advanced filtering expressions can be used to display only the required information. For details on the use and capabilities of *Filter windows*, see *Section 3.3.10, Filtering list entries (p. 48)*.

The lower section of the **Quarantine contents** window contains a command bar to manipulate the selected objects, and a preview box displaying the first 4 Kb of the file. The following options are available from the command bar:

- **View**: Display the entire file in a new window.
  Open with: Open the object with the specified application. The application will be started on the local machine running ZMC.

  Save as: Save the object to the local machine running ZMC.

Send as e-mail: Send the selected object(s) as e-mail attachment to the destination address specified in the appearing dialog window.

Forward as e-mail: Forward the selected e-mail to the destination address specified in the appearing dialog window. This option is available only to quarantined e-mails.

Delete: Delete the selected object(s).

Tip
Delete and **Send as e-mail** can be used at once on multiple selected objects.

In case of clusters, the command bar includes a node-selection combobox, that allows to display the contents of all nodes, or only a specified one.

14.3.1. Information stored about quarantined objects

The following meta-information is stored about the objects in the quarantine:

- **Client address**: It sets theIP address and the port of the client receiving the quarantined object.
- **Client zone**: It is the zone that the client belongs to.
- **Date**: It is the date when the object was quarantined.
- **Description**: It provides a detailed description of the verdict.
- **Direction**: It is the direction the quarantined object was transferred to (that is, *`upload`* or *`download`*).
- **Detected type**: It is MIME-type of the quarantined object as detected by ZCV.
- **File**: It is the file name or URL of the quarantined object.
- **File ID**: It defines a unique identifier of the file in the quarantine.
- **From**: It sets the sender address (in case of e-mails).
- **Group**: It is the user who tried to access the object belongs to the listed usergroups.
- **Kind**: It identifies the kind of the quarantined content: *`file`*, *`e-mail`*, or *`newsnet post`*.
- **Method**: It is the HTTP method (for example, *`GET`*, *`POST`*) in which the quarantined object was detected.
- **Program**: It defines the program that quarantined the object (usually ZCV or Zorp).
- **Protocol**: It sets the protocol in which the quarantined object was found.
- **Proxy**: It is the name of the proxy class that requested content vectoring on the quarantined object.
- **Recipient**: It is the envelope recipient addresses of the object (only in SMTP).
- **Reason**: It describes the reason why the object was quarantined (for example, detected as virus, spam, and so on).
- **Rule group**: It is the ZCV rule group that was stacked by the proxy.
- **Scanpath**: It sets the scanpath that quarantined the object.

- **Sender**: It is the envelope sender address of the object (only in SMTP).
- **Server address**: It identifies the IP address and the port of the server sending the quarantined object.
- **Server zone**: It sets the zone that the server belongs to.
- **Session ID**: It is the ID of the session which requested content vectoring on the quarantined object.
- **Size**: It defines the Size of the object in bytes.
- **Spam status**: It indicates if the e-mail is detected as spam.
- **Subject**: It describes the subject of the e-mail.
- **To**: It is the recipient address (in case of e-mails).
- **Type**: It defines the MIME-type of the quarantined object according to its MIME header.
- **User**: It identifies the name of the user who tried to access (for example, download) the object.
- **Verdict**: It is the decision that caused the object to be quarantined (for example, *`REJECT`*, *`ACCEPT_QUARANTINE`*, and so on)
- **Viruses**: It describes the virus(es) detected in the object.

Naturally, only the information relevant to the specific object is available, for example, an infected file downloaded through HTTP does not have subject, and so on.

14.3.2. Configuring quarantine cleanup

Quarantine cleanup on a host can be configured from the **Quarantine** tab of the given **Host** component in ZMC. This interface can be used to create rules that determine when objects are deleted from the quarantine.

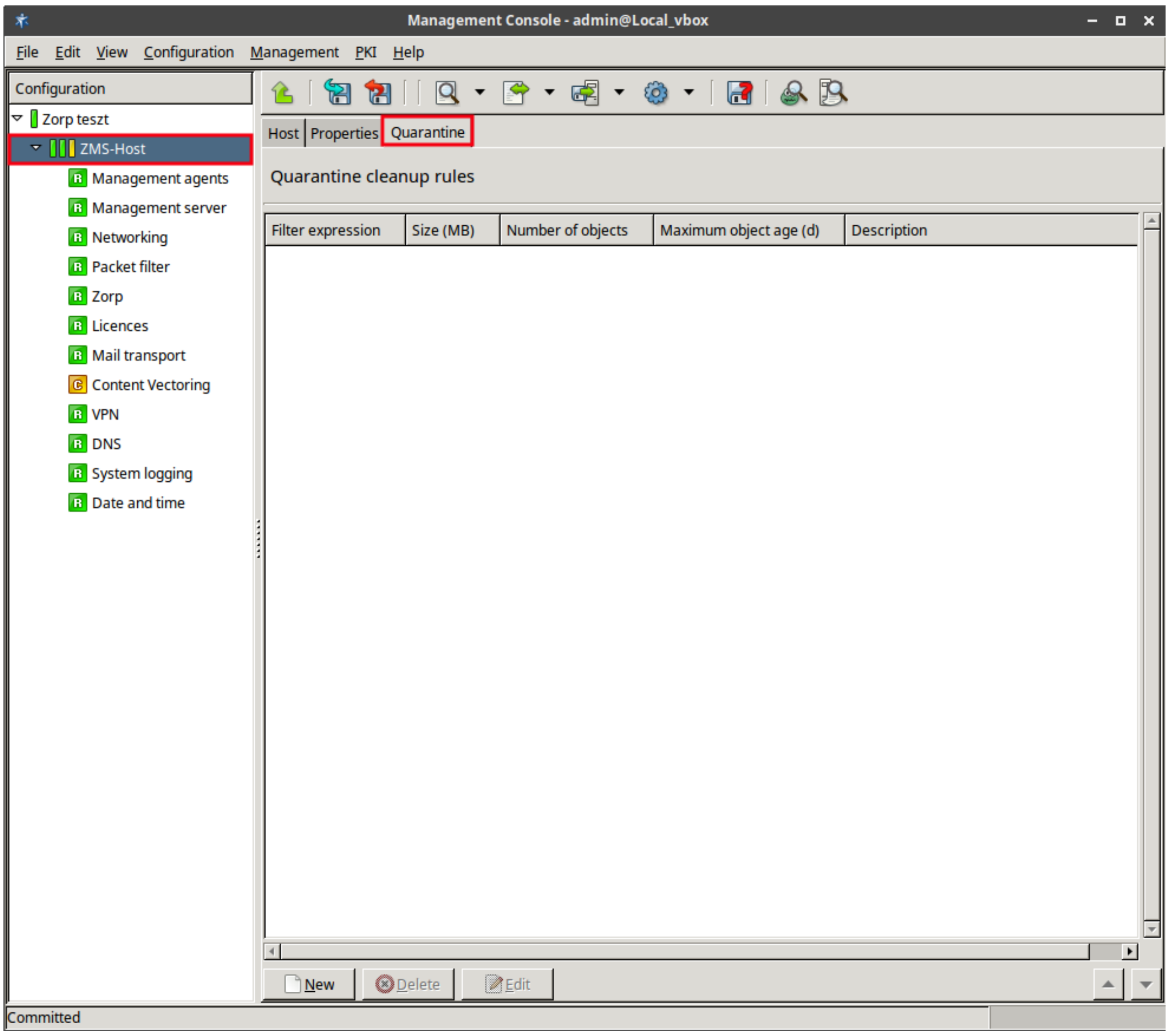
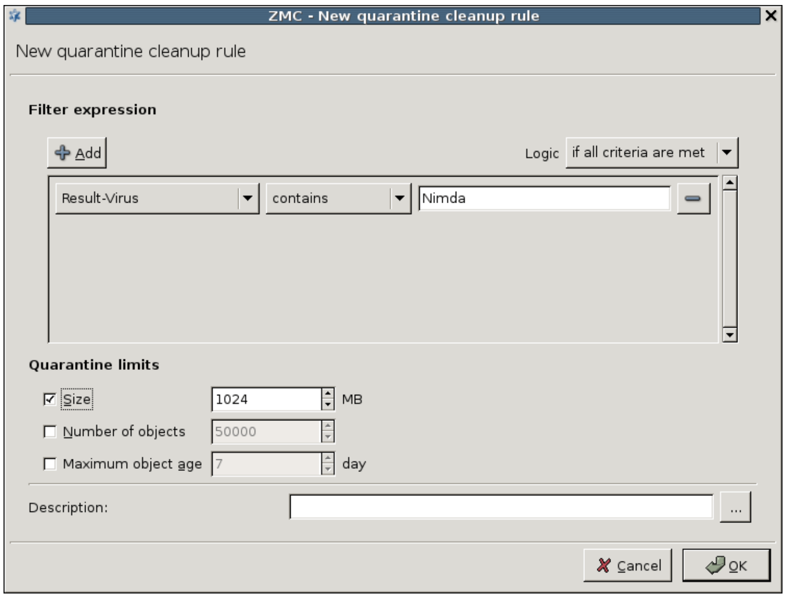


Figure 14.33. Configuring quarantine cleanup rules

The main section of the tab displays the currently effective rules (including disabled ones), and the control buttons for managing (creating, deleting, and editing) them. A rule consists of a filter that determines the scope (the effected objects) of the rule, and limitations on the storage of such objects. The available filtering expressions are the same as used in the **Advanced** filter optons of the **Quarantine contents** panel (see *Section 14.3, Quarantine management in ZMC (p. 384)*). The following options can be used to set limitations on the stored objects:

- **Size**: It defines the maximum hard disk space used to store the objects. Objects exceeding this limit are deleted (starting with the oldest object).
- **Number of objects**: It sets the maximum number of stored objects. Objects exceeding this limit are deleted (starting with the oldest object).
- **Maximum object age**: The objects older than the specified value are deleted (starting with the oldest object).


ZMC - New quarantine cleanup rule
New quarantine cleanup rule
Filter expression
Add
Cancel
OK

Logic: if all criteria are met

Result-Virus contains Nimda

Quarantine limits

☑ Size 1024 MB
☐ Number of objects 50000
☐ Maximum object age 7 day

Description:

Figure 14.34. Creating new cleanup rules

The limitations only apply to the objects matching the set filter expression. For example, setting the filter expression to `Result-Virus matches X` and the **Size** limit to 256 MBytes means that a maximum of 256 MBytes of objects will be stored in the quarantine that are infected with the X virus.

Tip
The limitations are global if no filter is specified.

Rules are evaluated and executed sequentially; if contradicting rules are found, the strictest one will be effective.

Chapter 15. Connection authentication and authorization

User authentication verifies the identity of the user trying to access a particular network service. When performed on the connection level, it enables the full auditing of the network traffic. Authentication is often used in conjunction with authorization — allowing access to a service only to clients who have the right to do so.

15.1. Authentication and authorization basics

Authentication is a method to ensure that certain services (access to a server, and so on) can be used only by the clients allowed to access the service. The process generally called as authentication actually consists of three distinct steps:

- *Identification*: Determining the clients identity (for example, requesting a username).
- *Authentication*: Verifying the clients identity (for example, requesting a password that only the real client knows).
- *Authorization*: Granting access to the service (for example, verifying that the authenticated client is allowed to access the service).

Note
It is important to note that although authentication and authorization are usually used together, they can also be used independently. Authentication verifies the identity of the client. There are situations where authentication is sufficient, because all users are allowed to access the services, only the event and the user's identity has to be logged. On the other hand, authorization is also possible without authentication, for example if access to a service is time-limited (for example, it can only be accessed outside the normal working hours, and so on). In such situations authentication is not needed.

Verifying the clients identity requires an authentication method based on something the client knows (for example, password, the response to a challenge, and so on), or what the client has (for example, a token, a certificate, and so on). Traditionally, firewalls authenticate the incoming connections based on the source IP of the connection: if a user has access (can log in) to that computer, he has the right to use the services. However, there are several problems with this approach. IP addresses can be easily forged (especially on the local network), and are not necessarily static (for example, when DHCP is used). Furthermore, this method cannot distinguish the different users who are using a single computer (for example, in a terminal server or hot-desking environment). For these reasons, authentication is most commonly left to the server application providing the particular service. However, Zorp is capable to overcome these problems in a simple, user-friendly way.

15.1.1. Inband authentication

Most protocols (for example, HTTP, FTP) capable of authentication offer only *inband* authentication, meaning that the client must authenticate himself on the server. The advantage of inband authentication is that it is an internal part of the protocol, and most client applications support it. The disadvantage is that many protocols do not support any form of authentication, and those that do, support only a few authentication methods. Usually in an organization it is desirable to use only a single (strong) authentication method, however, not all protocols are suitable for all methods.

Note
A few protocols support authentication on the firewall as well, in this case the client actually has to authenticate himself twice: once on the firewall, once on the server.

15.1.2. Outband authentication

Outband authentication is performed independently of the service and the protocol in a separate communication channel. Consequently, any protocol can be authenticated, and the authentication method does not depend on the protocol. That way every protocol can be authenticated with a single authentication method. The only disadvantage of outband authentication is that a special client application (for example, the Zorp Authentication Agent) has to be installed and configured on all client machines.

The process of outband authentication using the authentication agent is illustrated on the figure below.

15.1.2.1. Procedure – Outband authentication using the Zorp Authentication Agent

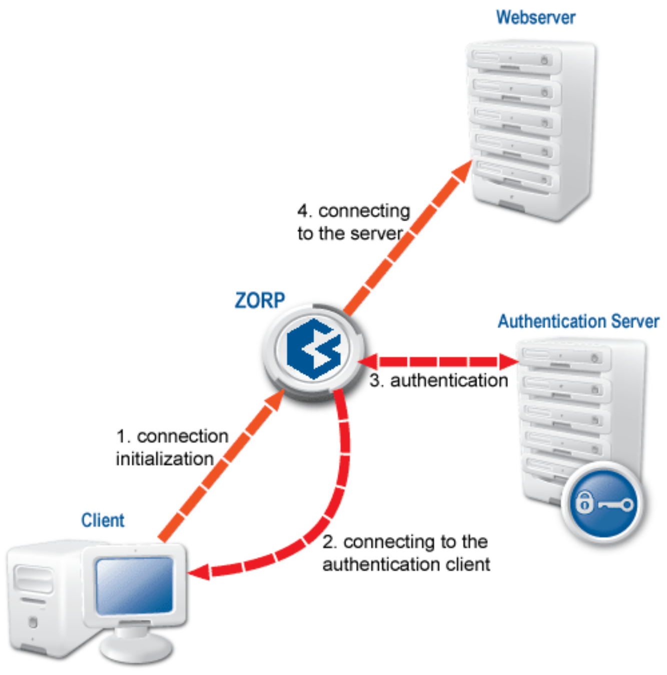

Webserver
4. connecting to the server
ZORP
Authentication Server
3. authentication
1. connection initialization
Client
2. connecting to the authentication client

Figure 15.1. Outband authentication in Zorp

Step 1. The client tries to connect to the server.

Step 2. Zorp connects to the authentication agent of the client.

Step 3. The actual authentication is performed:

Step a. username

Step b. authentication method selection

Step c. authentication according to the selected method

Step 4. If the authentication is successful, the connection to the server is established.

15.2. The concept of ZAS

ZAS is a tool that allows Zorp services to be authenticated against existing user databases (backends). ZAS does not itself provide authentication services, it only mediates between Zorp and the backends. The traditional access control model in Zorp is based on verifying that a connection requesting a service from a source zone is allowed to access the service, and that the service is allowed to enter the destination zone. That is:

- The client must belong to a zone where the particular service can be initiated (outbound service).
- The service must be allowed to enter (inbound service) the destination zone.

Using ZAS to authenticate the connections adds further requirements to this model: the client must successfully authenticate himself, and (optionally) must be allowed to access the service (that is, authorization can be also required). The actual procedure is as follows:

When the client initiates a connection, it actually tries to use a Zorp service. Zorp checks if an *authentication policy* is associated to the service. If an *authentication policy* is present, Zorp contacts a ZAS server (the *authentication provider* specified in the *authentication policy*). The ZAS server can connect to a user database (a *backend*) storing user information (passwords, certificates, and so on) required for a particular authentication method. (The type of the database determines which authentication methods can be used.) Each *instance* of a ZAS server can connect to a single *backend*, but multiple *instances* can be run from ZAS. When an authentication request arrives from Zorp, ZAS evaluates a number of configured *routers*: rules that determine which *instance* should be used to authenticate the connection. This decision is based on meta-information of the connection received from Zorp (for example, `Client-IP`, `Username`, and so on). The selected *instance* connects the client and the authentication is performed. The authentication can take place within the protocol (inband), or using a dedicated authentication agent (outband).

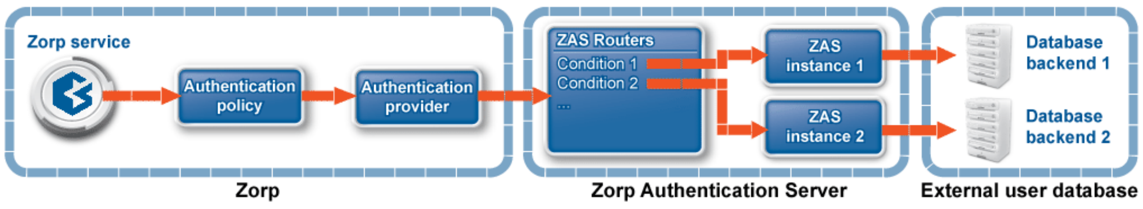


Figure 15.2. The operation of ZAS

If the authentication is successful, Zorp verifies that the client is allowed to access the service (by evaluating the *authorization policy*). If the client is authorized to access the service, the server-side connection is built. The client is automatically authorized if no authorization policy is assigned to the service.

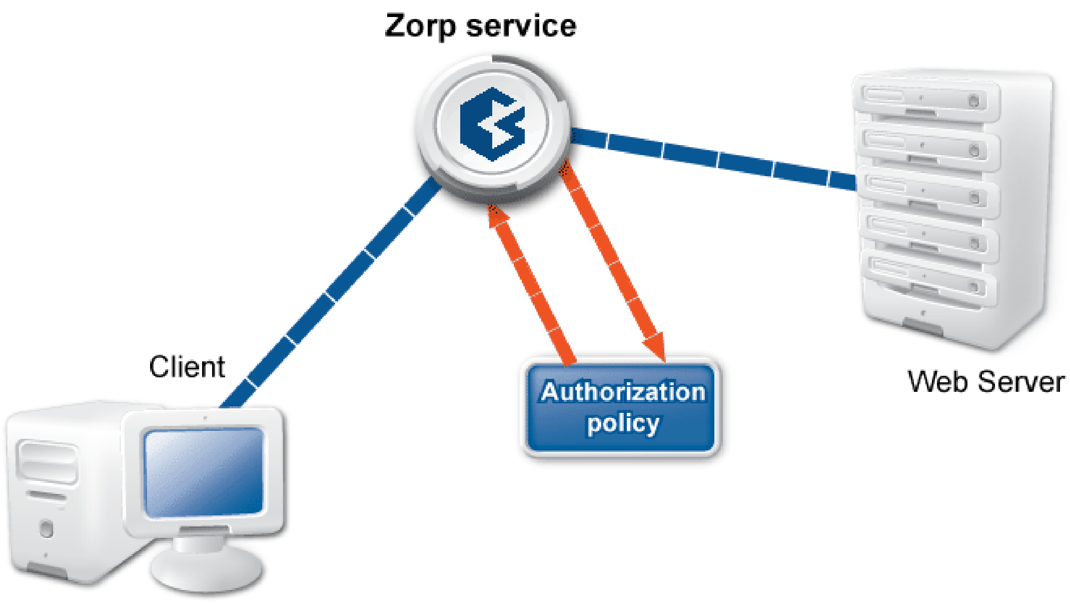


Figure 15.3. Authorization in Zorp

15.2.1. Supported backends and authentication methods

ZAS currently supports the following database backends:

- ***zas_db***: zas_db authenticates users against an LDAP database, supporting the following authentication methods: *`username/password`*, *`S/Key`*, *`CryptoCard RB1`*, *`LDAP binding`*, *`GSSAPI/Kerberos5`*, and *`x.509`*.
- ***file***: It is an authentication through the traditional `htpasswd` file. it supports the *`username/password`* authentication method.
- ***PAM***: It means an authentication using Pluggable Authentication Modules. Any authentication method supported in PAM can be used.
- ***RADIUS***: It is an authentication using a RADIUS server. It supports the *`username/password`* and *`challange/response`* authentication methods.

15.3. Authenticating connections with ZAS

This section describes how to initialize and configure ZAS, and how to create the required Zorp policies.

15.3.1. Configuring ZAS

The various parameters of ZAS can be configured on the **Authentication Server** ZMC component. Before starting to configure ZAS, add this component to the host (see *Procedure 3.2.1.3.1, Adding new configuration components to host (p. 22)* for details).

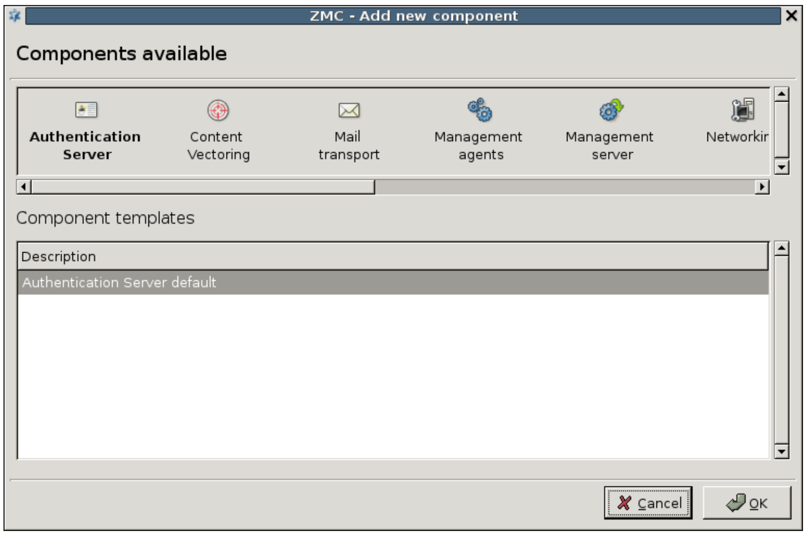


Figure 15.4. Adding the Authentication server component to the host

15.3.1.1. Configuring backends

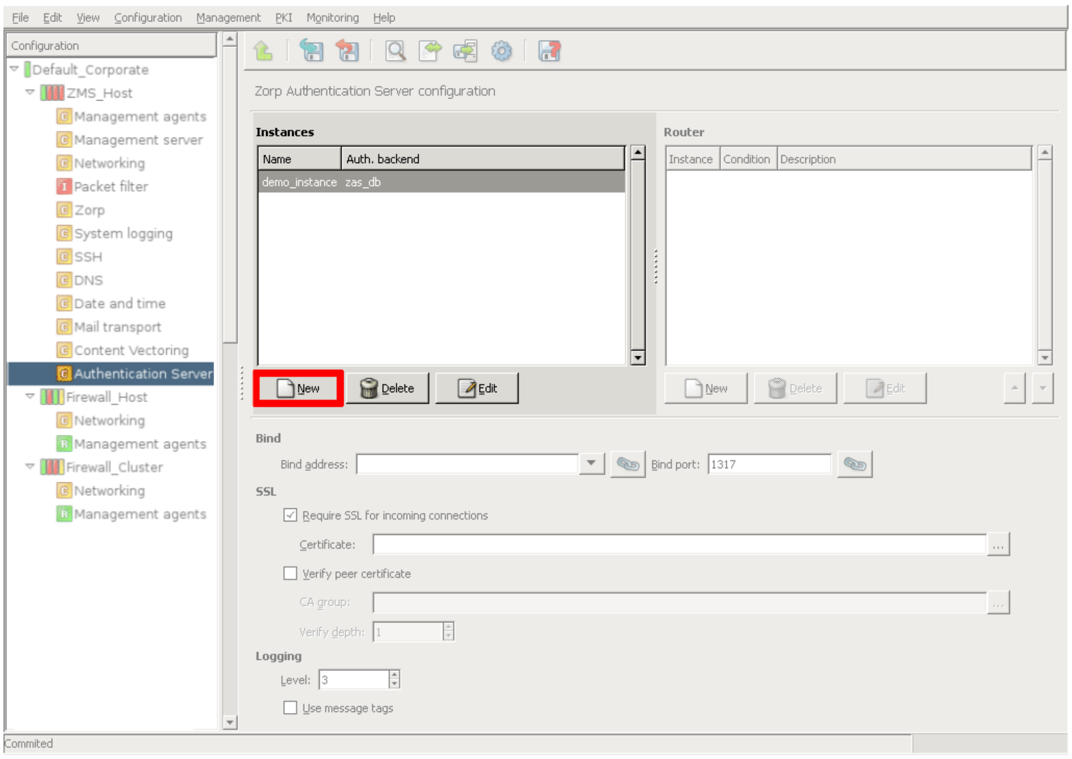
ZAS instances using specific database backends can be configured in the **Instances** section of the **Authentication Server** ZMC component. The existing instances and the type of database they use are displayed in a list; instances can be created, deleted, and modified using the control buttons below the list.

Note
Only unused instances can be deleted; if an instance is used in a router, the router has to be modified or deleted first.

To create a new instance, complete the following procedure:

15.3.1.1.1. Procedure – Creating a new instance

Step 1. Navigate to the **Authentication Server** ZMC component, and click on **New** in the **Instances** section.



Name	Auth. backend
demo_instance	zas_db

Bind

Bind address: Bind port: 1317

SSL

☑ Require SSL for incoming connections

Certificate:

☐ Verify peer certificate

CA group:

Verify depth: 1

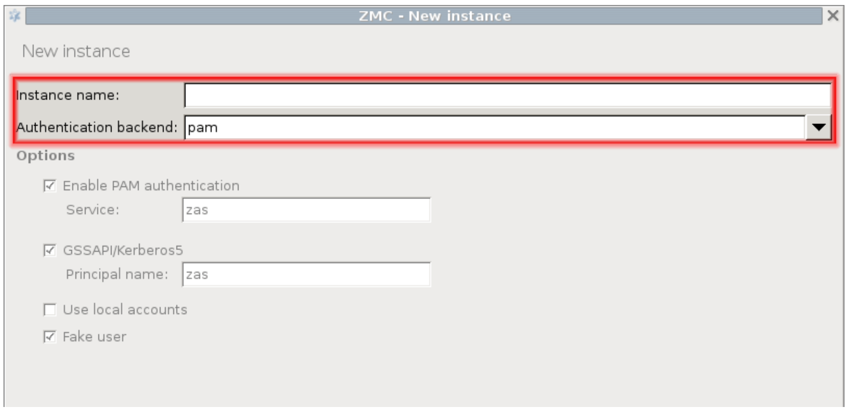
Logging

Level: 3

☐ Use message tags

Figure 15.5. Creating a new instance

Step 2. Enter a name for the instance and select the type of the database this instance will connect to from the **Authentication backend** combobox. Options specific to the selected backend type will be displayed.



Instance name:

Authentication backend: pam

Options

☑ Enable PAM authentication

Service: zas

☑ GSSAPI/Kerberos5

Principal name: zas

☐ Use local accounts

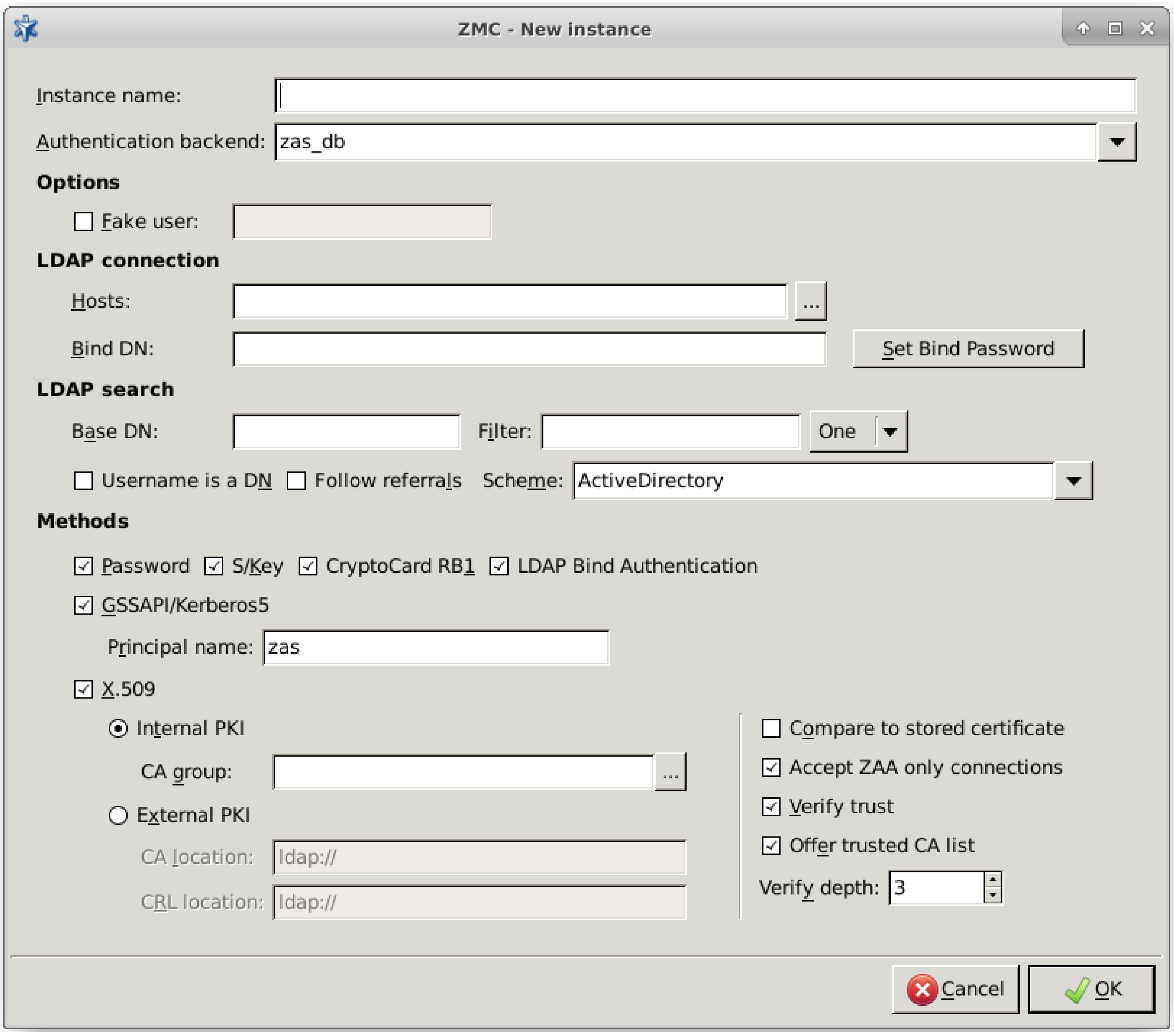
☑ Fake user

Figure 15.6. Selecting backend type

Step 3. Configure the options of the backend. The available backends and their options are described in the following chapters. The permitted authentication methods can be also selected here.

The zas_db backend

The *zas_db* backend authenticates users against an LDAP database using the Microsoft Active Directory, the POSIX, or the Novell eDirectory/NDS scheme.


ZMC - New instance

Instance name:

Authentication backend: zas_db

Options

☐ Fake user:

LDAP connection

Hosts:

Bind DN:

Set Bind Password

LDAP search

Base DN: Filter: One

☐ Username is a DN ☐ Follow referrals Scheme: ActiveDirectory

Methods

☑ Password ☑ S/Key ☑ CryptoCard RB1 ☑ LDAP Bind Authentication

☑ GSSAPI/Kerberos5

Principal name: zas

☑ X.509

● Internal PKI

CA group:

○ External PKI

CA location: ldap://

CRL location: ldap://

☐ Compare to stored certificate

☑ Accept ZAA only connections

☑ Verify trust

☑ Offer trusted CA list

Verify depth: 3

Cancel OK

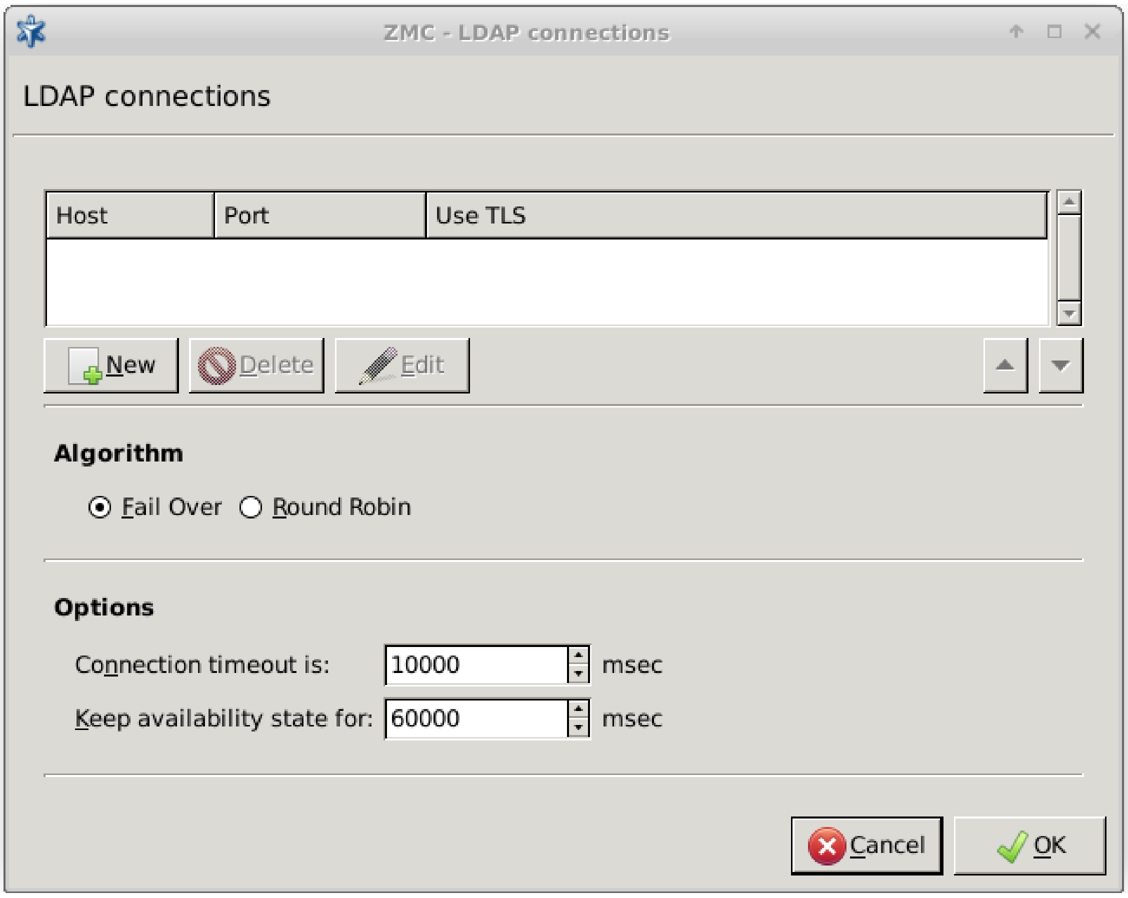
Figure 15.7. The zas_db backend

The backend has the following settings:

- **Fake user**: Enable authentication faking. This requires a valid user account in the LDAP database that is exclusively used for this purpose. The user name of this account has to be set in the corresponding textbox.

> **Note**
> All backends are capable of authentication faking. This is a method to hide the valid usernames, so that they cannot be guessed (for example using brute-force methods). If somebody tries to authenticate with a non-existing username, the attempt is not immediately rejected: the full authentication process is simulated (for example, password is requested, and so on), and rejected only at the end of the process. That way it is not possible to determine if the username itself was valid or not. It is highly recommended to enable this option.

- **LDAP connection settings**: By clicking on **...** next to the **Hosts entry** host related information can be specified.



LDAP connections

Host	Port	Use TLS

Algorithm

[✓ Fail Over] Round Robin

Options

Connection timeout is: 10000 msec

Keep availability state for: 60000 msec

Figure 15.8. The host configuration window

- **LDAP connections**: It is the list of LDAP servers. To set the order of servers, use the arrow buttons below the list.
  - **Host**: It is the IP address of the LDAP server.
  - **Port**: It is the port number of the LDAP server.
  - **Use TLS**: Enable TLS encryption to secure the communication between ZAS and the backend.
- **Algorithm**: ZAS attempts to connect to the next host in the ordered **LDAP connections** list based on the selected algorithm.
- **Connection timeout is**: Specifies connection timeout to be used when connecting to the target server.
- **Keep availability state for**: ZAS does not try to connect to an unreachable server until the time set in milliseconds in the **Keep availability state for** option expires.
- **Bind DN**: Bind to this DN before accessing the database.
- **Set Bind password**: It shall be the password to use when binding to LDAP.

- **LDAP search settings**:

- **Base DN**: Perform queries using this DN as base.
- **Filter**: Search for accounts using this filter expression.
- **Scope**: It specifies the scope of the search. *`base`*, *`sub`*, and *`one`* are acceptable values, specifying *`LDAP_SCOPE_BASE`*, *`LDAP_SCOPE_SUB`*, and *`LDAP_SCOPE_ONE`*, respectively.
- **Username is a DN**: It indicates that the incoming username is a fully qualified DN.
- **Follow referrals**: If this option is set, ZAS will respect the referral response from the LDAP server when looking up a user.
- **Scheme**: Specify LDAP scheme to use: *`Active Directory`*, *`POSIX`*, or *`NDS`* style directory layout.

Note
Make sure to set *`Scheme`* to *`Active Directory`* when using a Microsoft Active Directory server as a database backend.

- **Authentication methods**: Select and configure the allowed authentication methods.
  - **Password**: It implements password authentication. Allow password authentication only if the connection between Zorp and ZAS is secured (see *Section 15.3.2, Authentication of Zorp services with ZAS (p. 401)* for details).
  - **S/Key**: It is the S/Key-based authentication.
  - **CryptoCard RB1**: It defines cryptoCard RB1 hardware token based authentication.
  - **LDAP Bind**: It is authentication against the target LDAP server. Only password authentication is supported by this method, therefore it is only available if the connection between ZAS and Zorp is secured with SSL.
  - **GSSAPI/Kerberos5**: It defines GSSAPI-based authentication. The **Principal name** representing this authentication service also has to be set.
  - **x.509**: It is authentication based on x.509 certificates. To use this method, a number of further options have to be specified:
    - **The CA issuing the client certificates.** This can be an internal CA group (managed by the Zorp PKI, see *Chapter 11, Key and certificate management in Zorp (p. 249)* for details), or an external one. In the latter case the locations of the trusted CA certificates and the corresponding CRLs have to be set as space-separated lists of `file://` or `ldap://` URLs.
    - **Compare to stored certificate**: Compare the stored certificate bit-by-bit to the certificate supplied by the client. The authentication will fail when the certificates do not match, even if the new certificate is trusted by the CA.
    - **Verify trust**: Verify the validity of the certificate (that is, the certificate has to be issued by one of the trusted CAs and must not be revoked). This verification is independent from the **Compare to stored certificate**, so if both parameters are set, both conditions must be fulfilled to accept the certificate.
    - **Verify depth**: The maximum length of the verification chain.
    - **Offer trusted CA list**: Send a list of trusted certificates to the client to choose from to narrow the list of available certificates.

- **Accept only ZAA connections**: **This option is available only in Zorp 3.4 or later.** By default, ZAS accepts connections only from Zorp Authentication Agents (ZAA). Disable this option if a different client is used to authenticate on ZAS, for example, if a web-browser authenticates using a client-side certificate.
  Disabling this option works only with proxies that support inband authentication, for example, HTTP.

The htpass backend

The **htpass** backend authenticates users against an Apache *htpasswd* style password file. The name (including the path) of the file to be used has to be specified in the **Filename** textbox. Authentication faking can be enabled by selecting the **Fake user** checkbox.

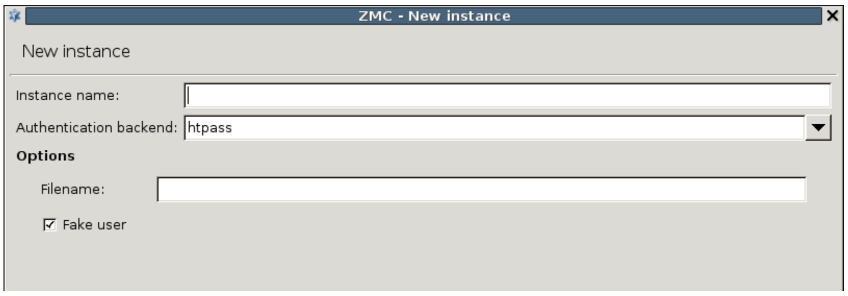

ZMC - New instance
New instance
Instance name:
Authentication backend: htpass
Options
Filename:
[✓] Fake user

Figure 15.9. The htpass backend

The Pluggable authentication module (PAM) backend

The PAM backend implements authentication based on the local authentication settings of the host running ZAS. It basically authenticates the users against the local PAM installation and/or using GSSAPI/Kerberos5.

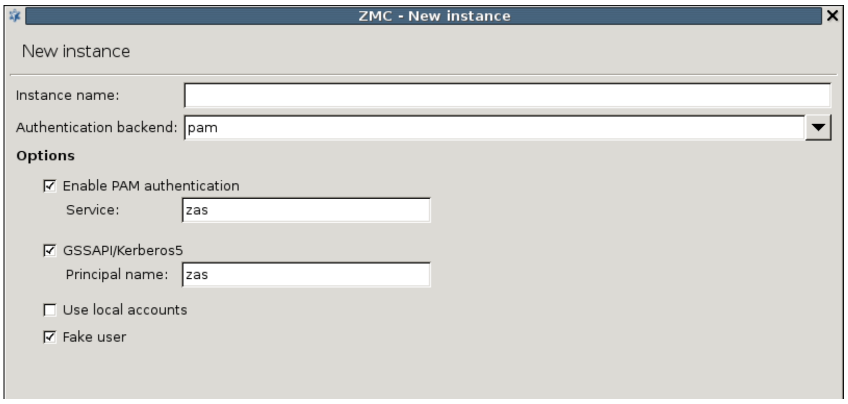

ZMC - New instance
New instance
Instance name:
Authentication backend: pam
Options
[✓] Enable PAM authentication
Service: zas
[✓] GSSAPI/Kerberos5
Principal name: zas
[ ] Use local accounts
[✓] Fake user

Figure 15.10. The PAM backend

The PAM backend has the following parameters:

- **Enable PAM authentication**: Enable PAM authentication. For PAM authentication the PAM service used for authentication has to be specified.
- **GSSAPI/Kerberos5**: Enable GSSAPI based authentication. The **Principal name** representing this authentication service also has to be set.

- **Use local accounts**: Use the local passwd/group database to query group membership of a given account.

Authentication faking can be enabled by selecting the **Fake user** checkbox.

The RADIUS backend

The RADIUS backend has the following parameters:

- **Host**: It is the hostname of the RADIUS server.
- **Port**: It is the port of the RADIUS server.
- **Secret**: It is the shared secret between the authentication server and ZAS.

Authentication faking can be enabled by selecting the **Fake user** checkbox.

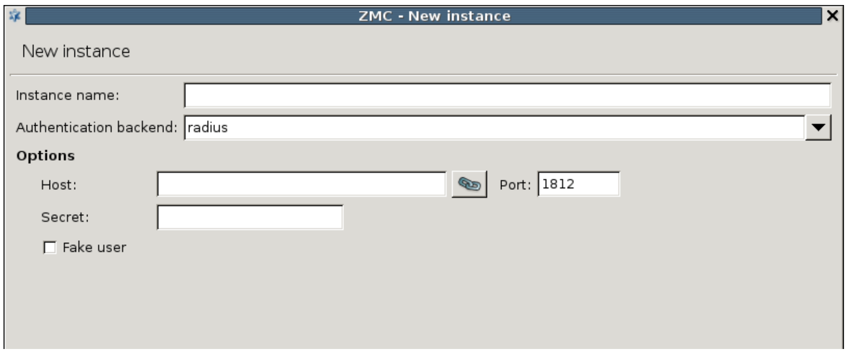


Figure 15.11. The RADIUS backend

15.3.1.2. Configuring routers

Routers are simple conditional rules (that is, if-then expressions) that determine which instance has to be used to authenticate a particular connection. They consist of a condition and a corresponding instance: if the parameter of the connection matches the set condition, then the authentication is performed with the set instance. The condition consists of a variable and a pattern: the condition is true if the variable of the connection is equal to the specified pattern. Routers can be configured in the **Routers** section of the **Authentication Server** ZMC component. They are evaluated sequentially: if the incoming connection matches a router, authentication is performed according to the instance specified in the router, otherwise the next router is evaluated. For configuring a new router only the condition has to be specified and the backend instance selected. The exact procedure is as follows:

1. Navigate to the **Authentication Server** ZMC component, and click **New** in the **Routers** section of the tab.

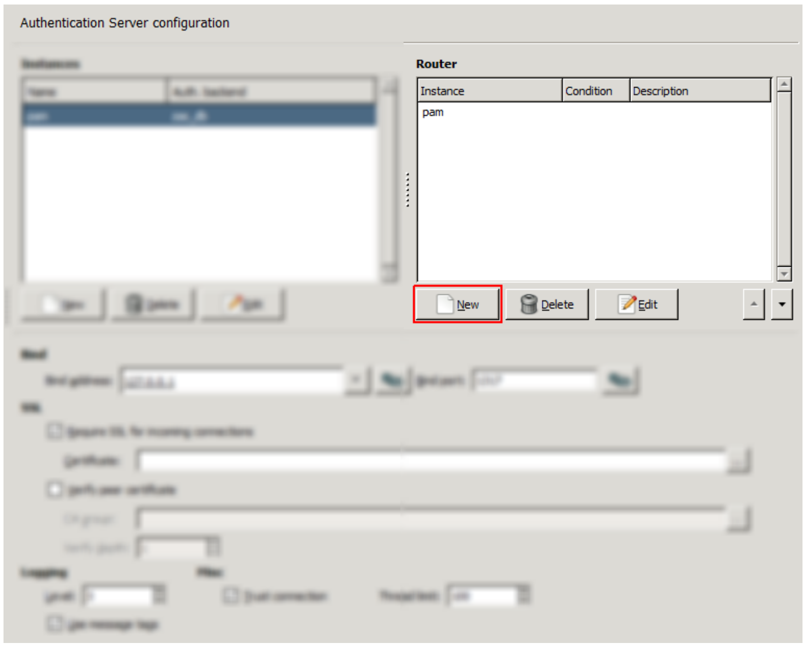
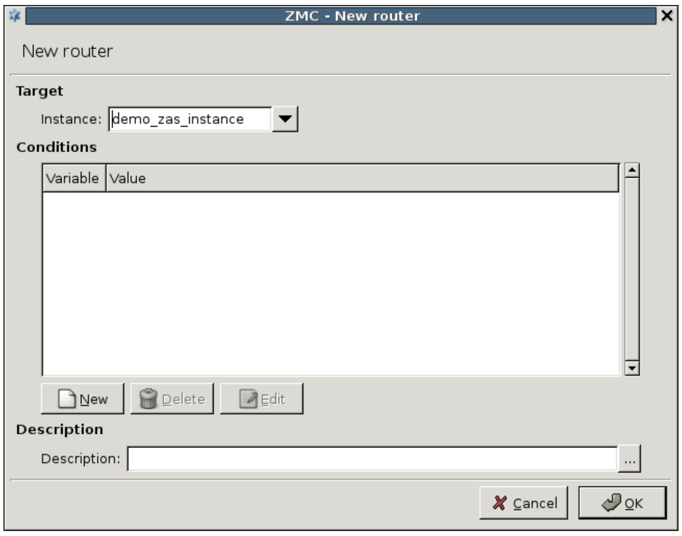


Figure 15.12. Defining new routers

2. Select the instance that will authenticate the connections matching this router from the **Target instance** combobox.

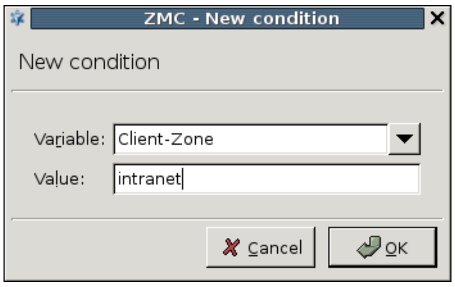


Instance: demo_zas_instance

Description:

Figure 15.13. Configuring a new router

3. Click on **New**, and define a condition for the router. Select the variable to be used from the **Variable** combobox, and enter the search term to the **Value** field. If the **Variable** of the inspected connection matches **Value**, the instance specified in **Target instance** will authenticate the connection.
   Currently the following variables can be used to create conditions: *`Client IP`*, *`Client zone`*, *`Service`*, and *`User`*.

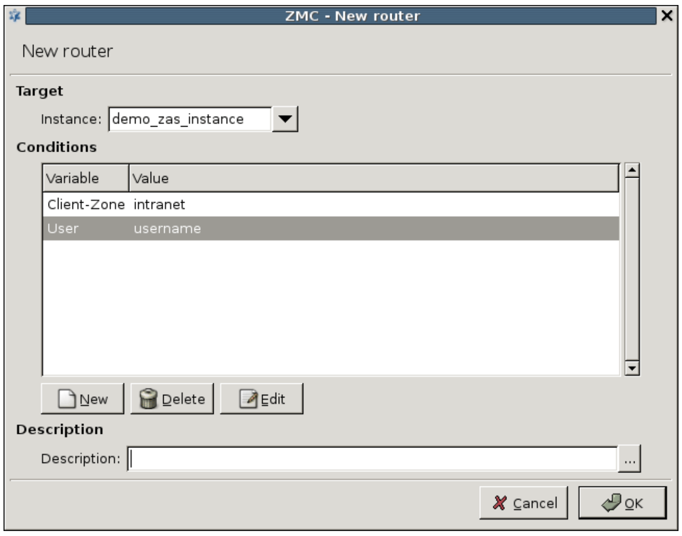


Variable: Client-Zone

Value: intranet

Figure 15.14. Defining conditions

Note
A router can contain multiple conditions. In this case all specified conditions must be true to select the target instance. (that is, the conditions are connected with logical AND operations.)



Instance: demo_zas_instance

Variable	Value
Client-Zone	intranet
User	username

Description:

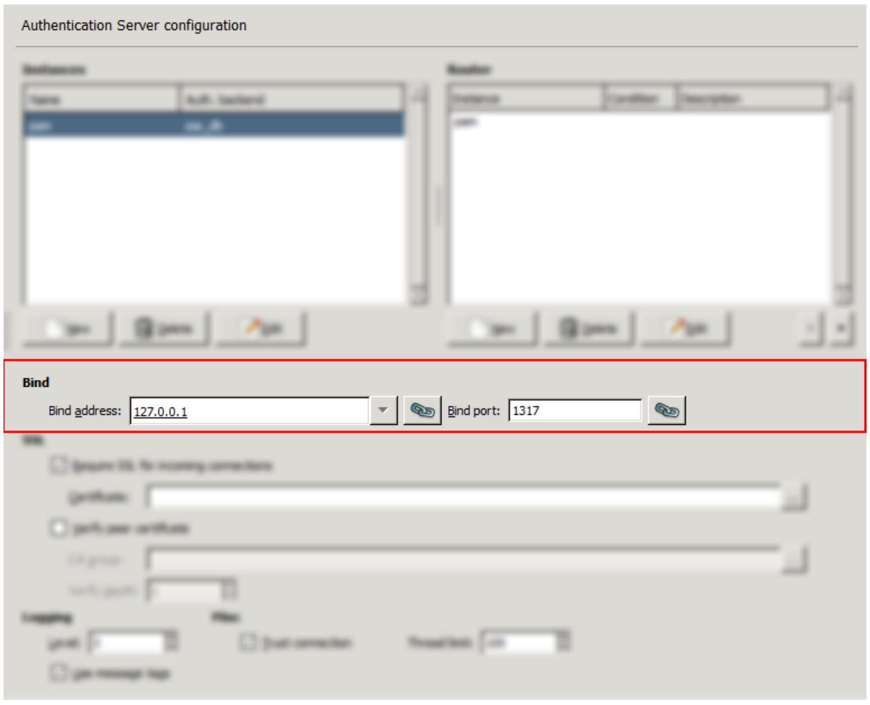
Figure 15.15. Using multiple conditions in a router

15.3.2. Authentication of Zorp services with ZAS

ZAS can only authenticate connections for which Zorp services explicitly request authentication. To allow this, the connection between Zorp and ZAS has to be set up. This requires configuring some connection parameters both in Zorp and in ZAS. The procedure below describes how to configure these parameters.

15.3.2.1. Procedure – Configuring communication between Zorp and ZAS

Step 1. First, the connection settings of ZAS have to be configured in the **Bind** section on the **Authentication server** ZMC component. Specify the IP address/port pair on which ZAS should accept connections.



Authentication Server configuration

Bind

Bind address: 127.0.0.1

Bind port: 1317

Figure 15.16. Configuring the bind parameters of ZAS

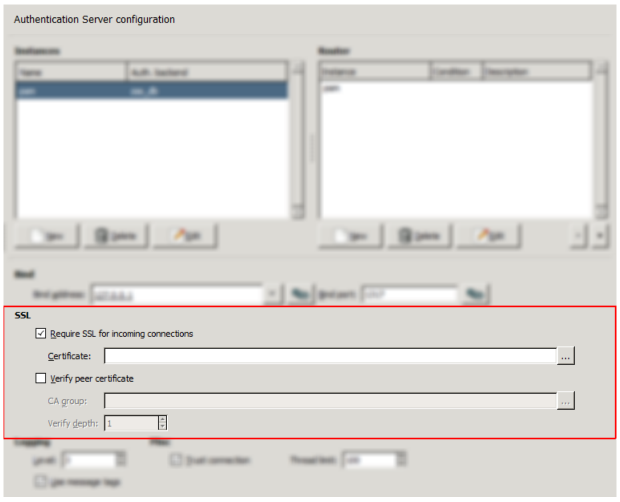
Tip

If ZAS and Zorp are running on the same machine, use the local loopback interface (IP:`127.0.0.1`).

Note

The same bind settings will have to be used when the Authentication provider is configured in the **Policies** tab of **Zorp** ZMC component.

Step 2. If Zorp and ZAS are running on separate machines, enable and configure SSL encryption. Check the **Require SSL for incoming connections** checkbox and click on **...** next to the **Certificate** textbox and select a certificate. This certificate has to be available on the ZAS host and will be presented to Zorp to verify the identity of the ZAS server. For details about creating certificates, see *Procedure 11.3.8.2, Creating certificates (p. 276)*.



Authentication Server configuration

SSL

☑ Require SSL for incoming connections

Certificate:

☐ Verify peer certificate

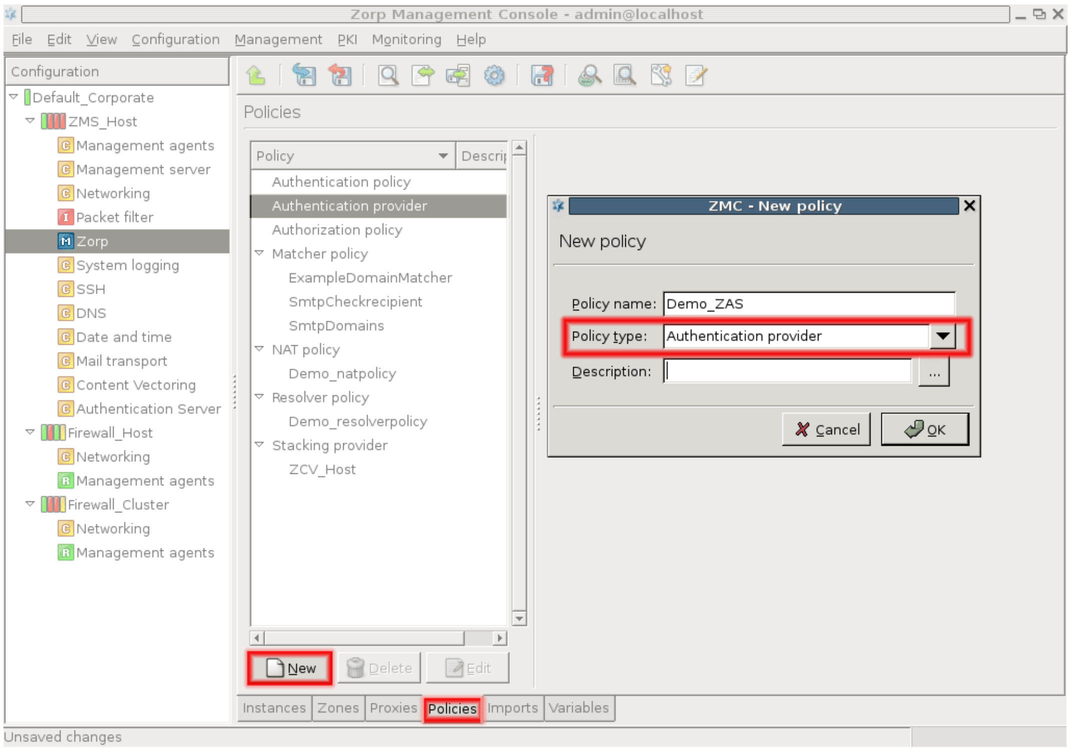
CA group:

Verify depth: 1

Figure 15.17. Configuring the SSL for ZAS

To enable mutual authentication (that is, to verify the certificate of Zorp), check the **Verify peer certificate** checkbox and select the **CA group** containing the trusted certificates. Also make sure to set the **Verify depth** high enough so that the root CA certificate in the CA chain can be verified. The default value (3) should be appropriate for internal CAs.

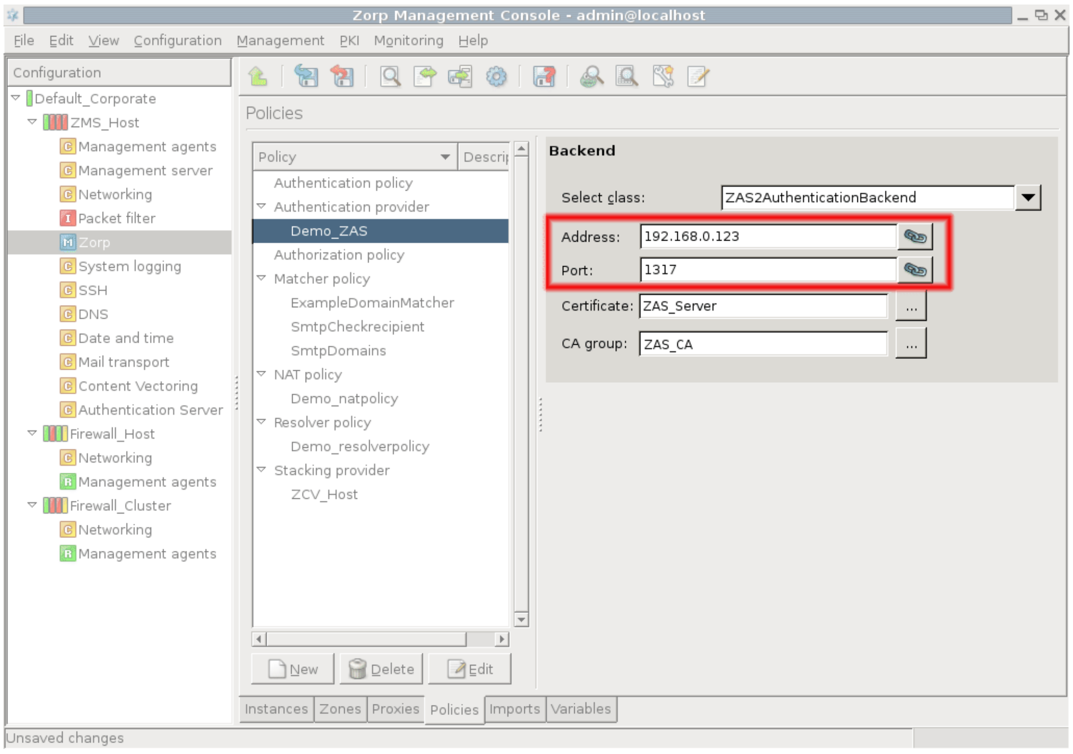
Step 3. The connection also has to be set up from the Zorp side. This can be accomplished by creating an **Authentication provider** on the **Policies** tab of the **Zorp** ZMC component. Click on **New**, select **Authentication provider** from the **Policy type** combobox, and enter a name for the provider into the **Policy** textbox.



Policy name: Demo_ZAS
Policy type: Authentication provider
Description:

Figure 15.18. Creating an Authentication provider

Step 4. Enter the IP address of the ZAS server into the **Address** field. This must be the same address as specified as **Bind address** for ZAS in Step 1.



Backend

Select class: ZAS2AuthenticationBackend

Address: 192.168.0.123

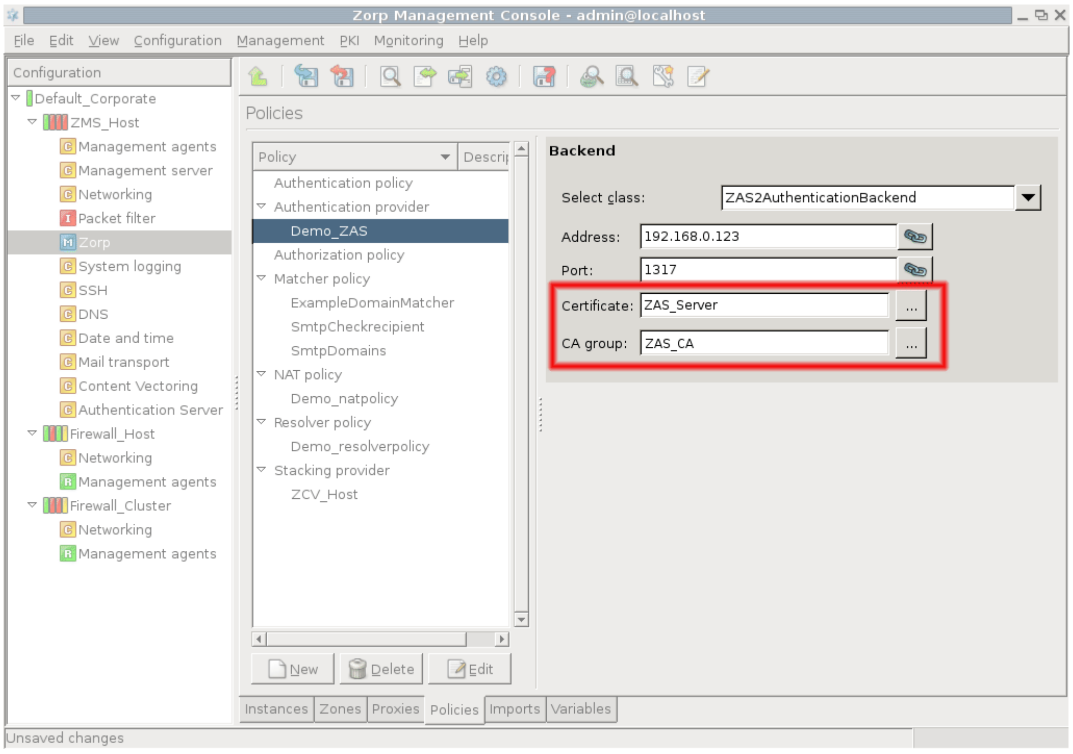
Port: 1317

Certificate: ZAS_Server

CA group: ZAS_CA

Figure 15.19. Configuring an Authentication provider

Step 5. If SSL encryption was enabled in Step 2, select the **Certificate** Zorp will show to ZAS. Zorp can also verify the certificate shown by ZAS using the CAs specified in **CA group**.



Select class: ZAS2AuthenticationBackend
Address: 192.168.0.123
Port: 1317
Certificate: ZAS_Server
CA group: ZAS_CA

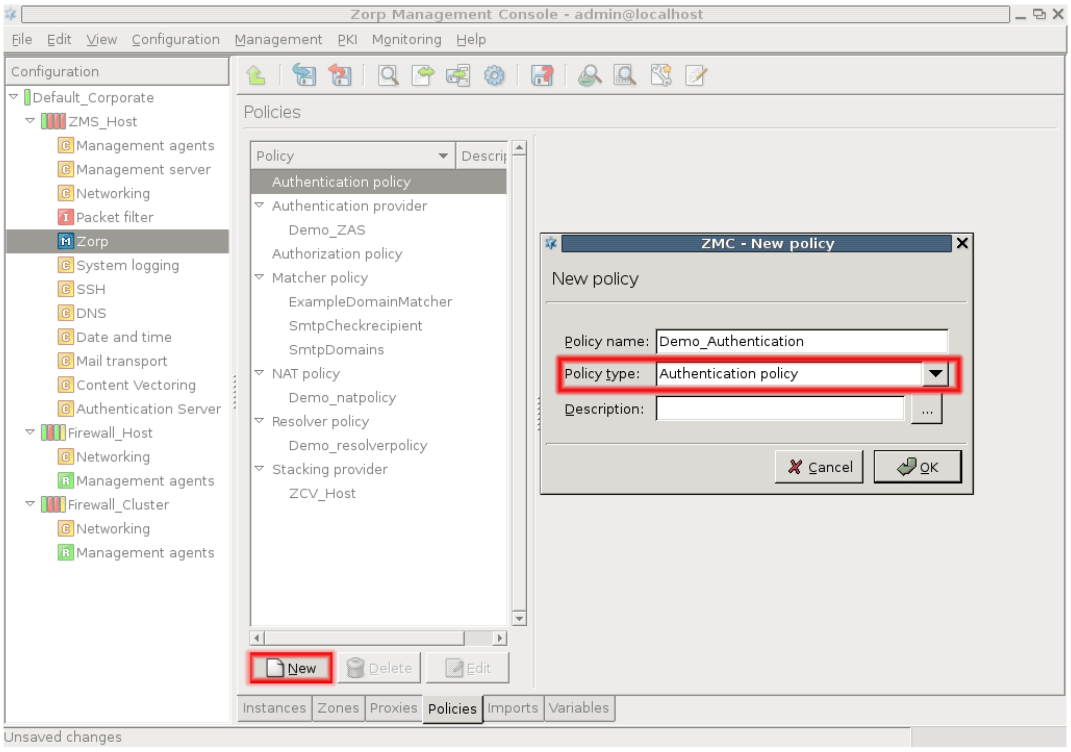
Figure 15.20. Configuring SSL for an Authentication provider

Note
Obviously, the CAs issuing the certificates of Zorp and ZAS must be members of the CA groups set to be used to perform the verification of the certificates, otherwise the verification will fail.

Now an **Authentication policy** has to be set up. Authentication policies are used by Zorp services and specify which authentication provider is used by the service, the type of authentication (inband, outband), and caching parameters. An authentication policy can be used by multiple services.

15.3.2.2. Procedure – Configuring Zorp Authentication policies

Step 1. Create an **Authentication policy** on the **Policies** tab of the **Zorp** ZMC component. Click on **New**, select **Authentication policy** from the **Policy type** combobox, and enter a name for the policy into the **Policy** textbox.



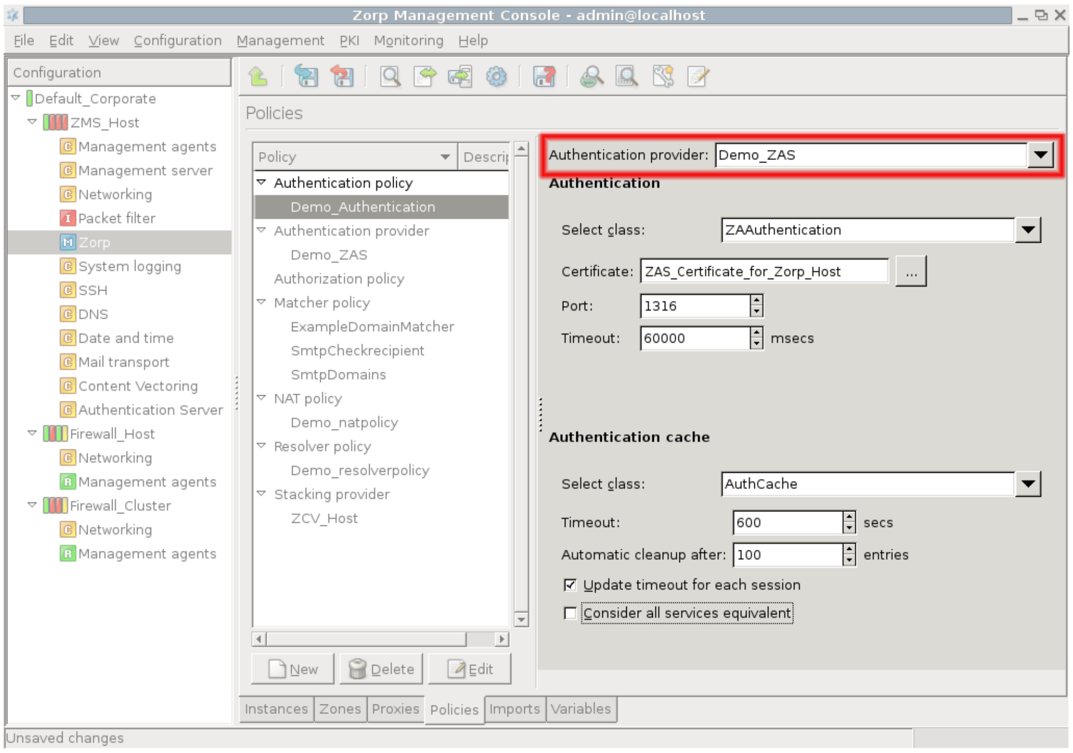
Policy name: Demo_Authentication

Policy type: Authentication policy

Description:

Figure 15.21. Creating Authentication policies

Step 2. Select the **Authentication provider** combobox by clicking **...** and selecting a provider.



Authentication provider: Demo_ZAS

Authentication

Select class: ZAAuthentication

Certificate: ZAS_Certificate_for_Zorp_Host

Port: 1316

Timeout: 60000 msecs

Authentication cache

Select class: AuthCache

Timeout: 600 secs

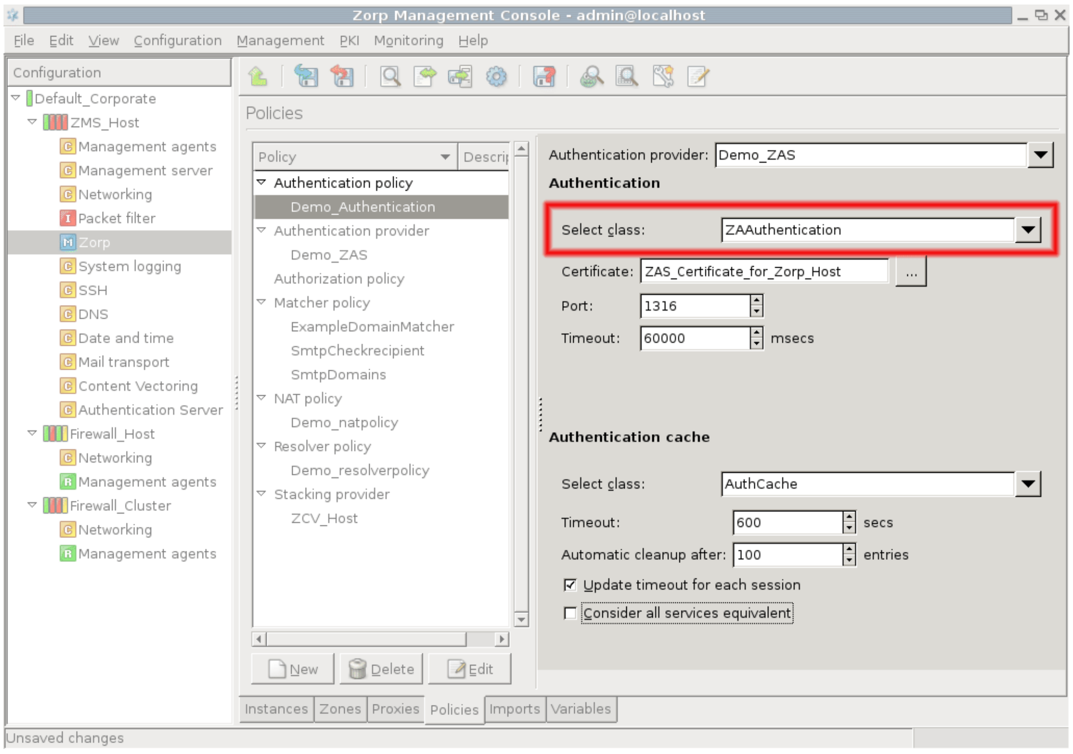
Automatic cleanup after: 100 entries

[✓] Update timeout for each session

[ ] Consider all services equivalent

Figure 15.22. Selecting the Authentication provider

Step 3. Select the type of authentication to be used from the **Class** combobox. The following authentication types are available:



Authentication provider: Demo_ZAS

Authentication

Select class: ZAAuthentication

Certificate: ZAS_Certificate_for_Zorp_Host

Port: 1316

Timeout: 60000 msecs

Authentication cache

Select class: AuthCache

Timeout: 600 secs

Automatic cleanup after: 100 entries

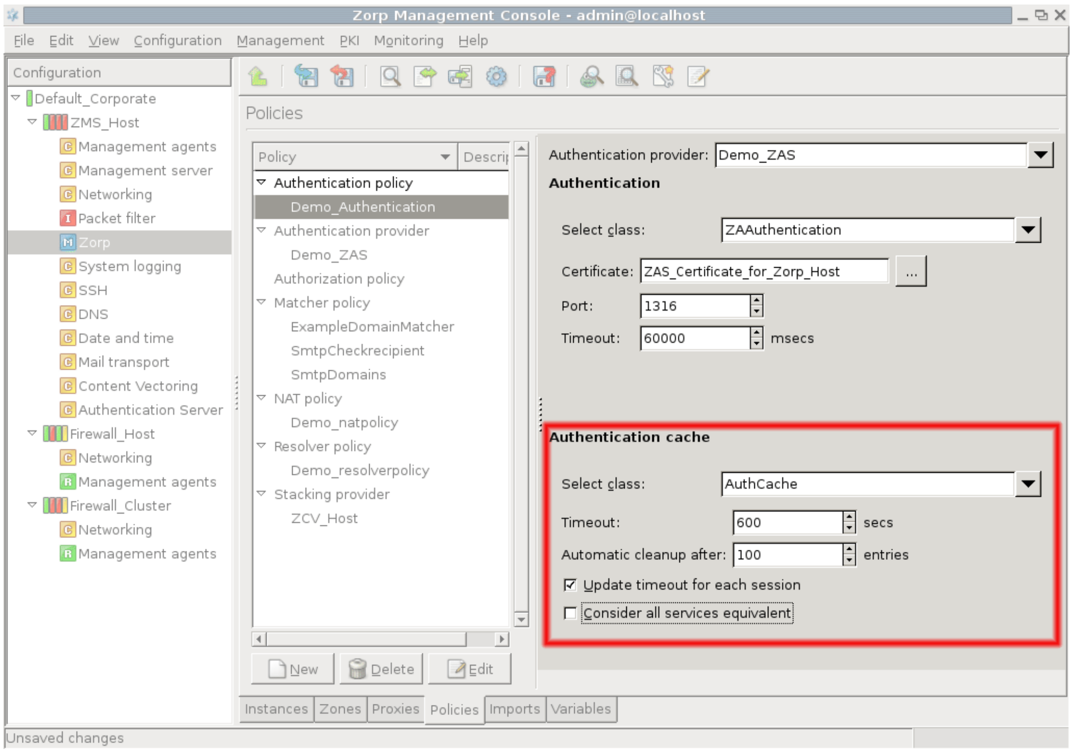
[✓] Update timeout for each session

[ ] Consider all services equivalent

Figure 15.23. Selecting the type of the authentication

- **Inband authentication**: Use the built-in authentication of the protocol to authenticate the client on Zorp.
- **ZAAuthentication**: (Also called Satyr authentication in previous versions). Outband authentication using the Zorp Authentication Agent. This method can authenticate any protocol. For agent authentication the following additional parameters have to be set:
  - **Certificate**: Select the certificate that Zorp will show to the authentication agent running on the client. The certificate is required because the communication between the authentication agent and Zorp is SSL-encrypted. The certificate has to be issued by a CA trusted by the authentication agent. The process of installing CA certificates for the authentication agent is described in *Chapter 6, Installing the Zorp Authentication Agent (ZAA)* in *Zorp Professional 7 Installation Guide*.
  - **Port**: The port where Zorp accepts connections from the authentication agents running on the clients.
  - **Timeout**: The period of time the client has to complete the authentication after an authentication request is sent by Zorp.
- **Server authentication**: Enable the client to connect to the target server, and extract its authentication information from the protocol.

Step 4. Configure the authentication cache using the **Class** combobox of the **Authentication cache** section. The following options are available:



Authentication provider: Demo_ZAS

Authentication

Select class: ZAAuthentication

Certificate: ZAS_Certificate_for_Zorp_Host

Port: 1316

Timeout: 60000 msecs

Authentication cache

Select class: AuthCache

Timeout: 600 secs

Automatic cleanup after: 100 entries

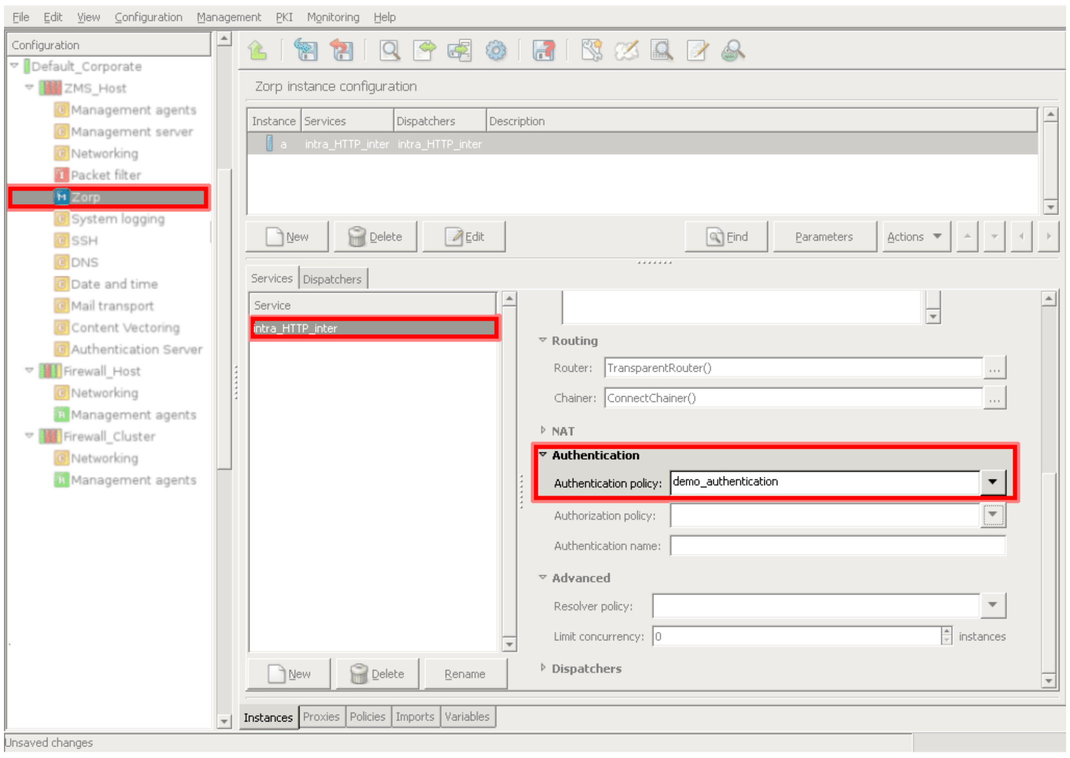
[✓] Update timeout for each session

[ ] Consider all services equivalent

Figure 15.24. Configuring the authentication cache

- **None**: Disable authentication caching. The client has to reauthenticate each time when starting a new service.
- **AuthCache**: Store the results of the authentication for the period specified in the **Timeout** field, that is, after a successful authentication the client can use the service (and start new ones of the same type) for that period. For example, once, being authenticated for an HTTP service, the client can browse the web for **Timeout** period, but has to authenticate again to use FTP.
  If the **Update timeout for each session** checkbox is selected, timeout measuring is restarted each time the client starts service. Selecting the **Consider all services equivalent** checkbox means that Zorp does not differentiate between the different services (protocols) used by the client, after a successful authentication he can use all available services without having to reauthenticate himself. For instance, if this option is enabled in the example above, the client does not have to reauthenticate for starting an FTP connection.

To actually use the authentication policy configured above, the Zorp services have to reference the policy.



Instance	Services	Dispatchers	Description
a	intra_HTTP_inter	intra_HTTP_inter	

Service: intra_HTTP_inter

Routing
Router: TransparentRouter()
Chainer: ConnectChainer()
NAT
Authentication
Authentication policy: demo_authentication
Authorization policy:
Authentication name:
Advanced
Resolver policy:
Limit concurrency: 0 instances
Dispatchers

Figure 15.25. Using authentication in Zorp services

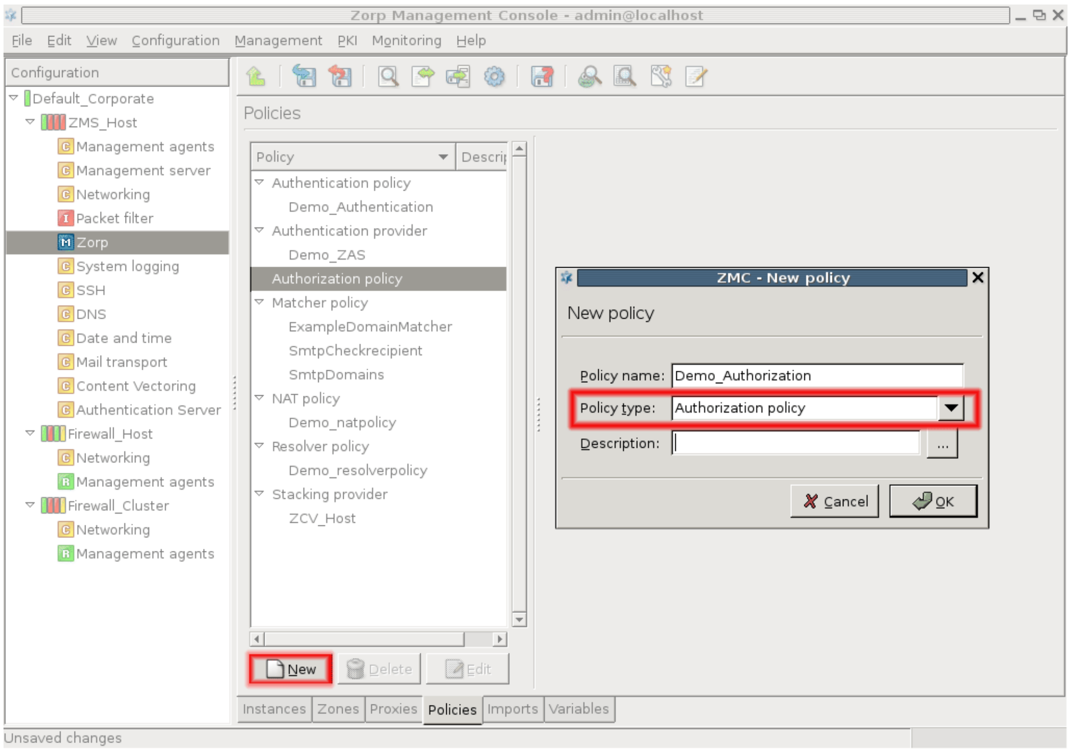
The authentication policy to be used by the service can be selected from the **Authentication policy** combobox on the **Instances** tab of the **Zorp** ZMC component. The combobox displays all the available authentication policies.

15.3.3. Authorization of Zorp services

Each Zorp service can use an *Authorization policy* to determine whether a client is allowed to access the service. If the authorization is based on the identity of the client, it takes place only after a successful authentication — identity-based authorization can be performed only if the client's identity is known and has been verified. The actual authorization is performed by Zorp, based on the authentication information received from ZAS or extracted from the protocol. Zorp offers various authorization models to ranging from simple (*`PermitUser`*) to advanced (*`NEyesAuthorization`*). Both identity-based and indentity-independent authorization models are available. The configuration of authorization policies is described in the procedure below.

15.3.3.1. Procedure – Configuring authorization policies

Step 1. Create an **Authorization policy** on the **Policies** tab of the **Zorp** ZMC component. Click on **New**, select **Authorization policy** from the **Policy type** combobox, and enter a name for the policy into the **Policy** textbox.



Policy name: Demo_Authorization

Policy type: Authorization policy

Description:

Figure 15.26. Creating authorization policies

Step 2. Select the authorization model to use in the policy from the **Class** combobox. The following models are available:

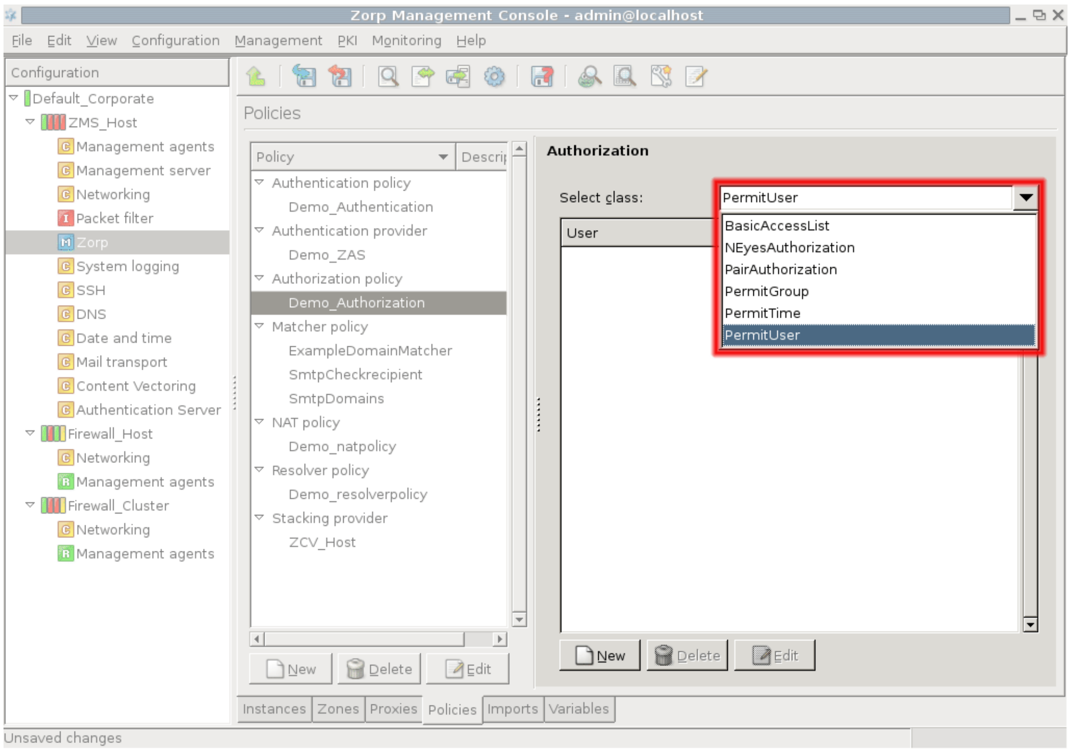


Figure 15.27. Selecting an authorization model

- ***BasicAccessList***: Authorize only users meeting a set of authorization conditions, for example, certain users, users belonging to specified groups, or any combination of conditions using the other authorization models.
- ***NEyesAuthentication***: The client trying to access the service has to be authorized by one (or more) authorized clients. This model can be used to implement 4-eyes authorization solutions.
- ***PairAuthentication***: Authorize only userpairs — single users cannot access a service, that is, only two different users (with different usernames) can access the service.

Tip
`NEyesAuthentication` and *`PairAuthentication`* are useful when the controlled access to sensitive (for example, financial) data has to be ensured and audited.

- ***PermitGroup***: Authorize only the members of the listed usergroups. This is a simplified version of the `BasicAccessList` model.
- ***PermitUser***: Authorize only the listed users. This is a simplified version of the `BasicAccessList` model.
- ***PermitTime***: Authorize any user but only in the set time interval. This authorization model does not require authentication.

Tip

Use the `BasicAccessList` authorization model to combine user authentication with time-based authentication. For example, create a policy consisting of two `Required` policies: PermitTime and PermitUser.

Step 3. Configure the parameters of the selected authorization class. See *Section 15.3.3.2, Authorization models of Zorp (p. 415)* for the detailed description of the classes.

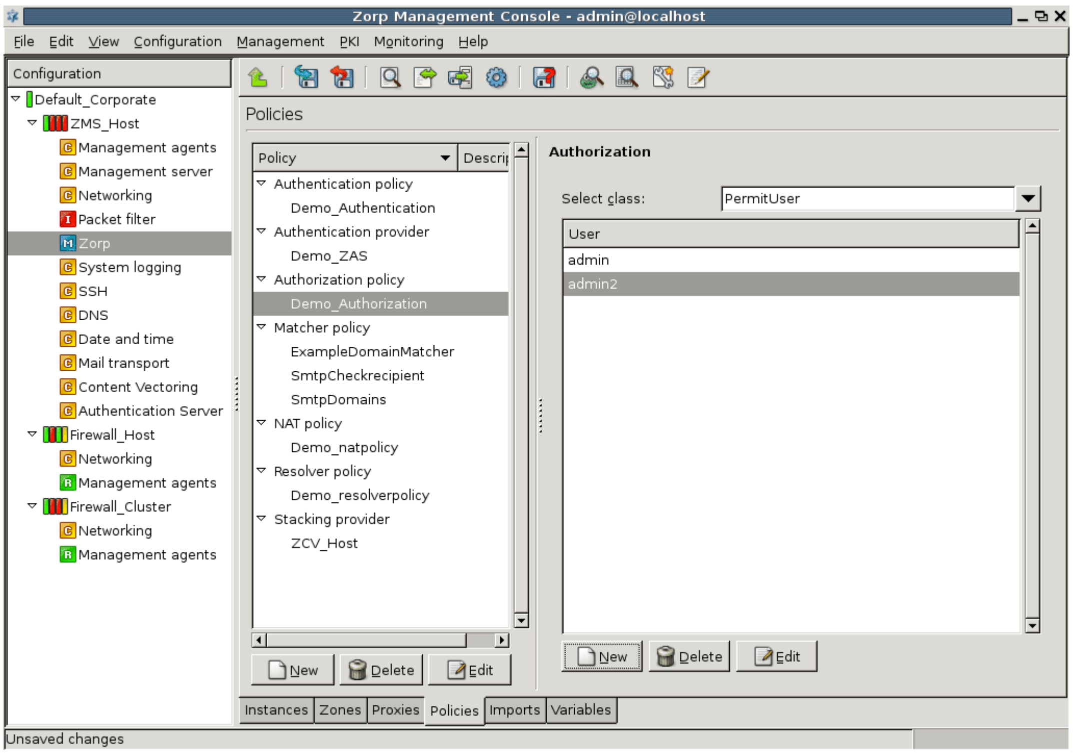
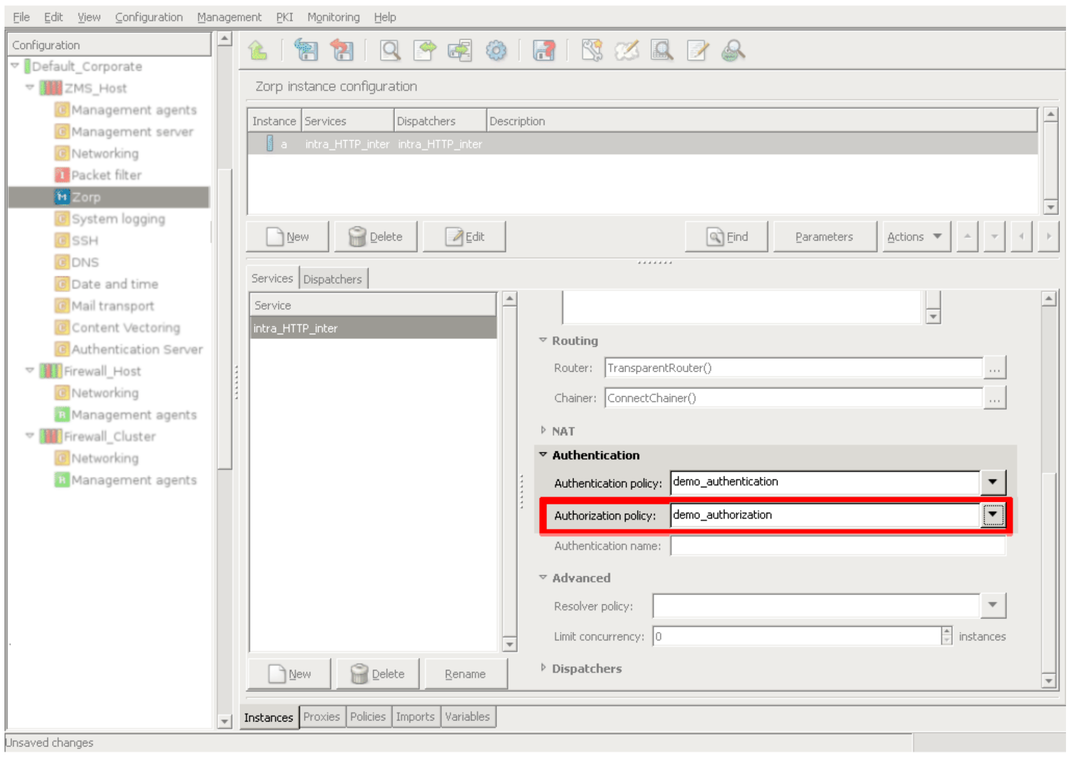


Figure 15.28. Configuring authorization policies

Step 4. Navigate to the **Instances** tab of the **Zorp** ZMC component, and select the service that will use the authorization policy.



Instance	Services	Dispatchers	Description
a	intra_HTTP_inter	intra_HTTP_inter	

Service: intra_HTTP_inter

Routing
Router: TransparentRouter()
Chainer: ConnectChainer()

NAT

Authentication
Authentication policy: demo_authentication
Authorization policy: demo_authorization
Authentication name:

Advanced
Resolver policy:
Limit concurrency: 0 instances

Dispatchers

Figure 15.29. Using authorization policies in Zorp services

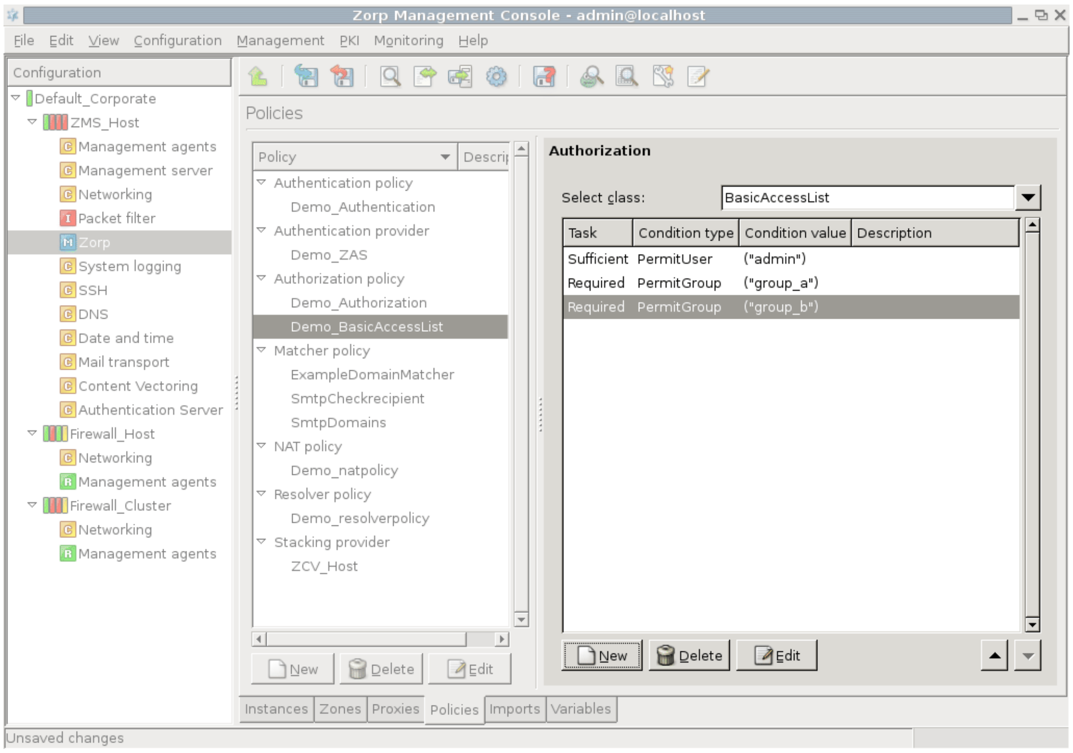
Step 5. In the **Service parameters** section, select the **Authorization policy** to use from the combobox.

15.3.3.2. Authorization models of Zorp

The configuration parameters of the authorization models available in Zorp are described in this section.

BasicAccessList

BasicAccessList can be used to create complex authorization scenarios by combining other authorization types into a set of *Required* or *Sufficient* conditions. Each condition refers to an authorization type (for example, *PermitUser*, *PairAuthorization*, and so on) and **Authentication task** (*Sufficient or Required*). The conditions are evaluated sequentially. A connection is authorized if a *Sufficient* condition matches the connection, or all *Required* conditions are fulfilled. If a *Required* condition is not met, the connection is refused.

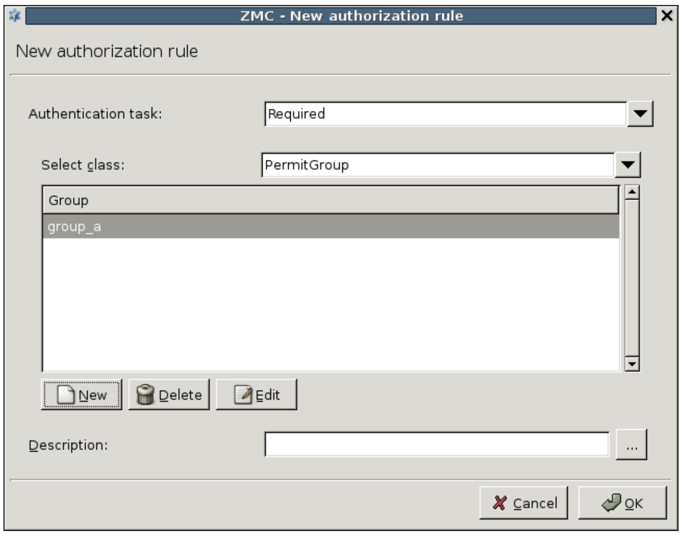

Zorp Management Console - admin@localhost
File Edit View Configuration Management PKI Monitoring Help
Configuration
Policies
Authorization
Instances Zones Proxies Policies Imports Variables
Unsaved changes

Select class: BasicAccessList

Task	Condition type	Condition value	Description
Sufficient	PermitUser	("admin")	
Required	PermitGroup	("group_a")	
Required	PermitGroup	("group_b")	

Figure 15.30. Using BasicAccessList

Note
Due to the sequential evaluation of the conditions, `Sufficient` conditions shall be placed to the top of the list.



Authentication task: Required

Select class: PermitGroup

Group
group_a

Description:

Figure 15.31. Creating BasicAccessList conditions

To create a new condition click **New**, and select the type of the condition from the **Authentication task** and **Condition** comboboxes. Then click on **New**, and enter the value for the condition (for example, the username or the name of the group).

Example 15.1. BasicAccessList
The following condition list allows the *admin* user and the users who are members of both *group_a* and *group_b* to access the service:

- **Authentication task**: *Sufficient*, **Condition**: *User*, **Value**: *admin*
- **Authentication task**: *Required*, **Condition**: *Group*, **Value**: *group_a*
- **Authentication task**: *Required*, **Condition**: *Group*, **Value**: *group_b*

NEyes authorization

When *NEyesAuthorization* is used, the client trying to access the service has to be authorized by another (already authorized) client (this authorization chain can be expanded to multiple levels). *NEyesAuthorization* can only be used in conjunction with another *NEyesAuthorization* policy. One of them is the *authorizer* set to authorize the *authorized* policy.

In a simple 4-eyes scenario the *authorizer* policy points to the authorized policy in its *Authorization policy* parameter, and has its *Wait for other authorization policies to finish* parameter disabled. The *authorized* policy has an empty *Authorization policy* parameter (meaning that it is at a *lower* stage, at the end of an N-eyes chain), and has its *Wait for other authorization policies to finish* parameter enabled, meaning that it has to be authorized by another policy.

NEyesAuthorization has the following parameters:

- **authorize_policy**: The authorization policy authorized by the current *NEyesAuthorization* policy.
- **Wait for other authorization policies to finish**: If this parameter is set, the client has to be authorized by another client. If set to *FALSE*, the current client is at the top of an authorizing chain.

  Note
  When setting this parameter, consider the timeout value set in the client application used to access the server. There is no use in specifying a longer time here, as the clients will time out anyway.

- **Max time for the authorization to arrive**: The time (in milliseconds) Zorp will wait for the authorizing user to authorize the one accessing the service.

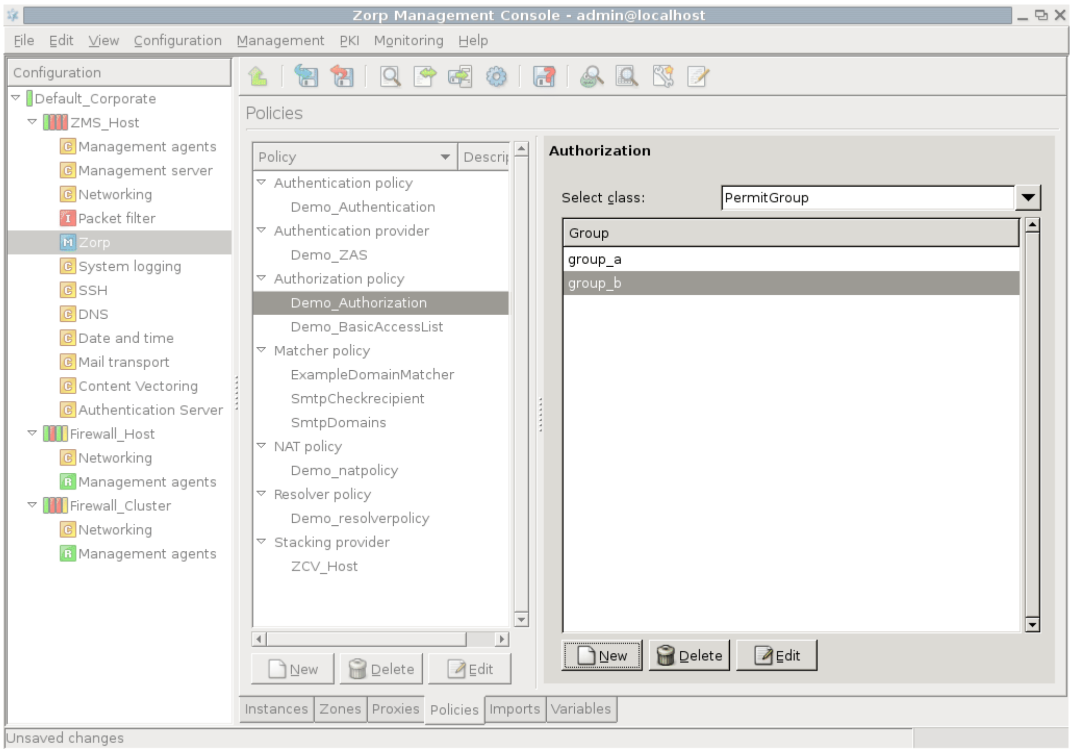
Pair authorization

When this authorization model is used, only two users simultaneously accessing the service are authorized, single users are not permitted to access the service. Set the time (in milliseconds) Zorp will wait for the second user to access the service using the *Max time for the pair to arrive* spinbutton.

When setting this parameter, consider the timeout valuse set in the client application used to access the server. There is no use in specifying a longer time here, as the clients will time out anyway.

PermitGroup

This model allows the members of the specified groups to access the service. Select the *grouplist* parameter and click **Edit**. In the appearing list editor window click **New**, then enter the name of an authorized usergroup. Additional groups can be added by clicking **New** again.



Select class: PermitGroup

Group
group_a
group_b

Figure 15.32. Using PermitGroup

Note
The elements of the list can be disabled through the local menu.

PermitUser

This model allows the listed users to access the service. Select the `userlist` parameter and click **Edit**. In the appearing list editor window click **New**, then enter the name of an authorized user. Additional users can be added by clicking **New** again.

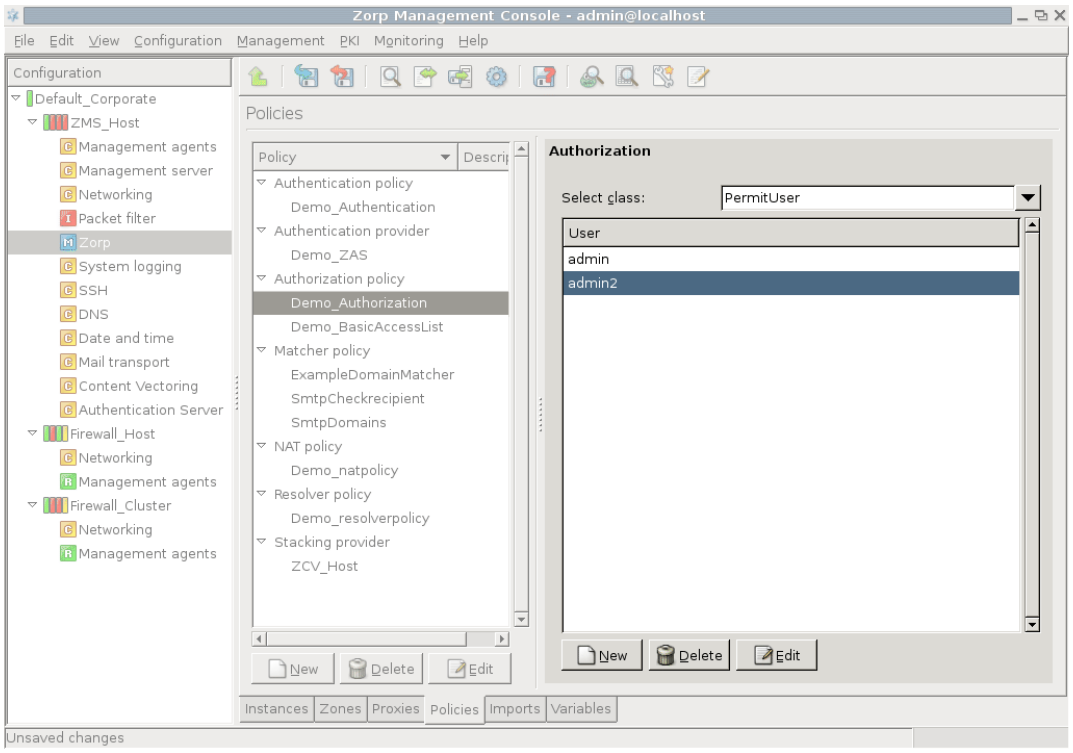


Figure 15.33. Using PermitUser

Note
The elements of the list can be disabled through the local menu.

PermitTime

The PermitTime policy stores a set of intervals specified by their starting and ending time — access to a service using such a policy is permitted only within this interval.

To configure an interval, click on **Edit**, then click **New**. Select the first `qstring` value and click on **Edit**. Enter the starting time of the interval (for example, *8:30*) and click **Ok**. The ending time of the interval can be set similarly through the second `qstring` value.

Note
A single policy can contain multiple intervals.

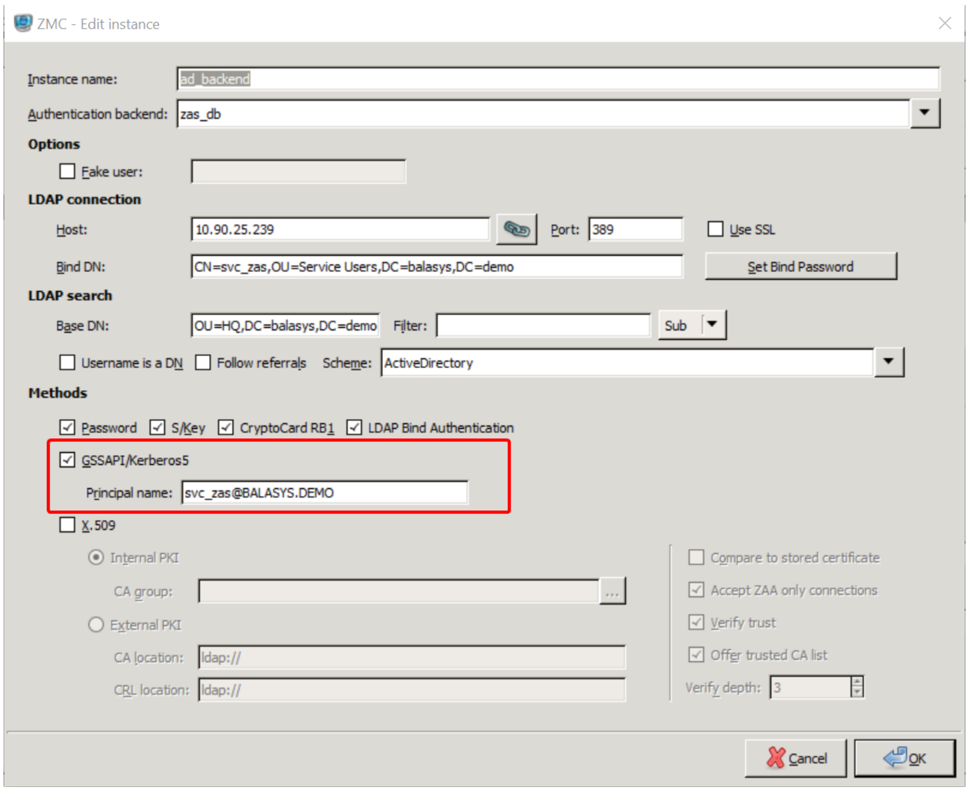
15.3.4. Procedure – Enabling Kerberos authentication in ZAS

Complete the following steps to enable Kerberos authentication in Zorp Authentication Server using Windows Active Directory (AD) environment.

Steps:

Step 1. In ZMC select **Authentication Server** > **Instances** > **Edit**.

Step 2. Select the **GSSAPI/Kerberos5** checkbox at **Methods** section and provide the *realm* at **Principal name** field.


ZMC - Edit instance

Instance name: ad_backend

Authentication backend: zas_db

Options

☐ Fake user:

LDAP connection

Host: 10.90.25.239 Port: 389 ☐ Use SSL

Bind DN: CN=svc_zas,OU=Service Users,DC=balasys,DC=demo

LDAP search

Base DN: OU=HQ,DC=balasys,DC=demo Filter: Sub

☐ Username is a DN ☐ Follow referrals Scheme: ActiveDirectory

Methods

☑ Password ☑ S/Key ☑ CryptoCard RB1 ☑ LDAP Bind Authentication

☑ GSSAPI/Kerberos5

Principal name: svc_zas@BALASYS.DEMO

☐ X.509

◉ Internal PKI

CA group:

○ External PKI

CA location: ldap://

CRL location: ldap://

☐ Compare to stored certificate

☑ Accept ZAA only connections

☑ Verify trust

☑ Offer trusted CA list

Verify depth: 3

Figure 15.34. Providing Kerberos realm

Step 3. Create the domain user in the **Active Directory**. Use the **Principal name** provided in the previous step.

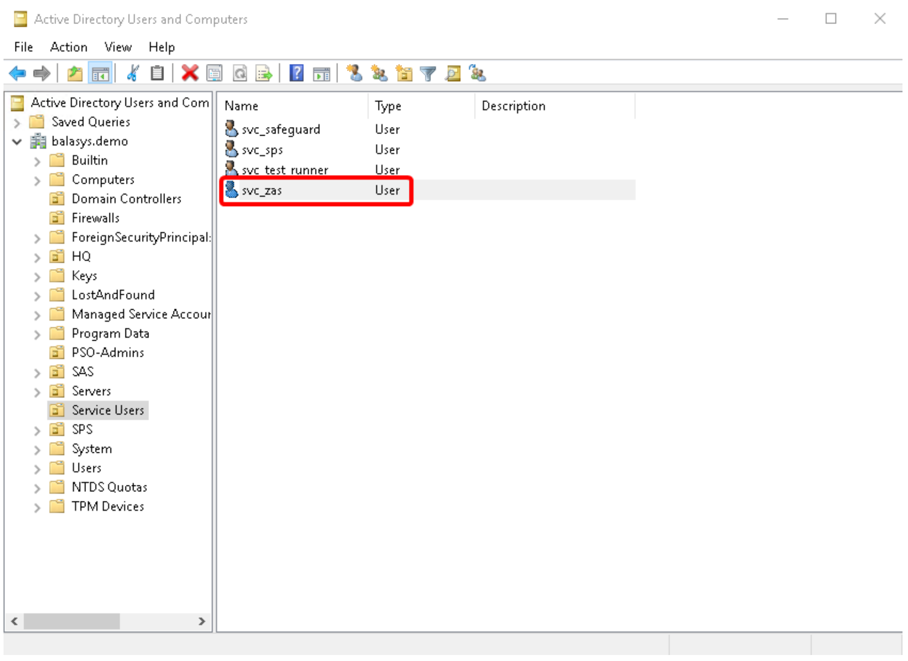
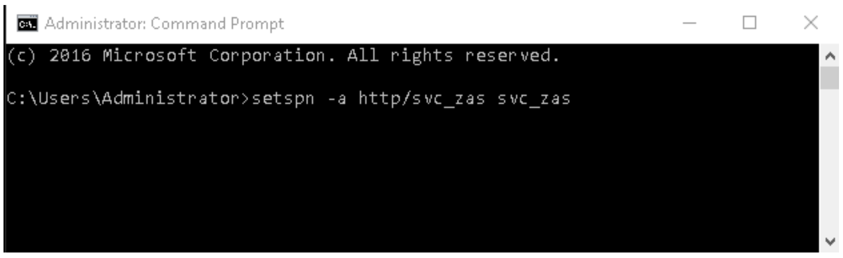


Figure 15.35. Creating the domain user

Step 4. Start the Command Prompt in the Domain Controller with Administrator privileges.

Step 5. Run the following command:

```
setspn -a http/ <username> <username>
```

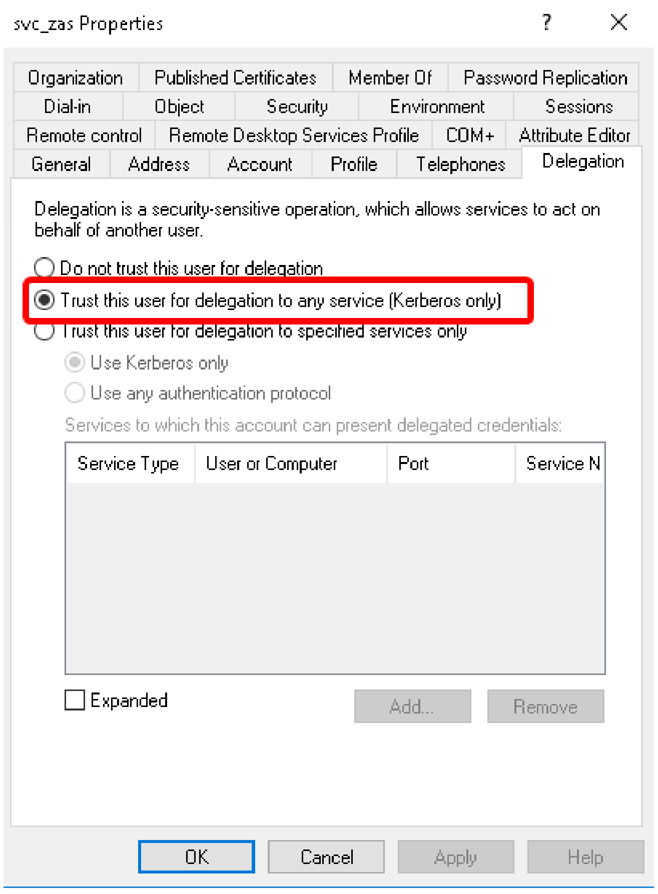


```
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>setspn -a http/svc_zas svc_zas
```

Figure 15.36. Running the command

Step 6. In the **Active Directory** window, select the user created in Step 3. and open the user’s **Properties**.

Step 7. A new **Delegation** tab is available now. Select the **Trust this user for delegation to any service (Kerberos only)** option. Click **Apply**.



Delegation is a security-sensitive operation, which allows services to act on behalf of another user.

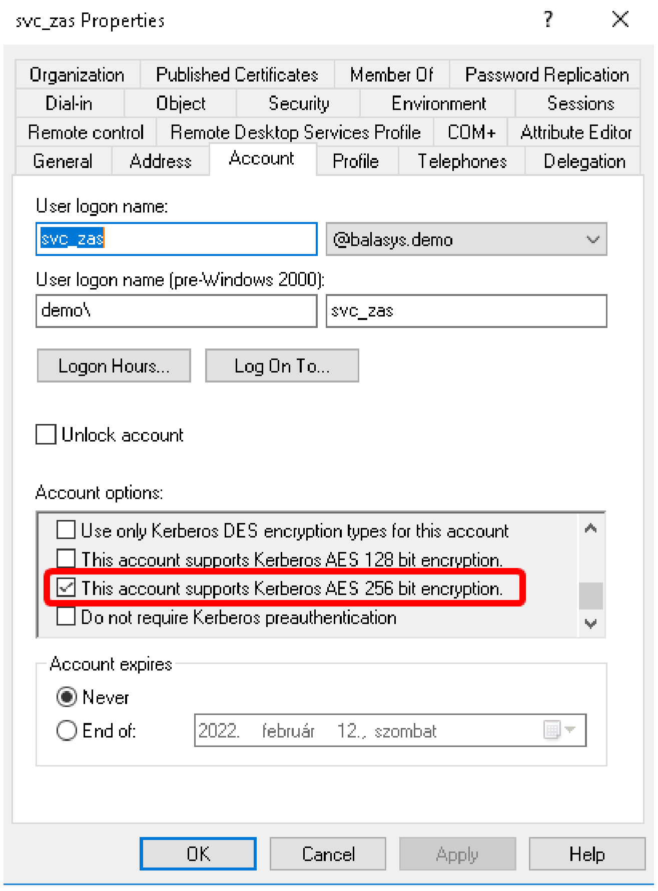
( ) Do not trust this user for delegation
(●) Trust this user for delegation to any service (Kerberos only)
( ) Trust this user for delegation to specified services only
(●) Use Kerberos only
( ) Use any authentication protocol

Services to which this account can present delegated credentials:

☐ Expanded

Figure 15.37. Authenticating a user

Step 8. Switch to the **Account** tab in the **Properties** menu item. Select the **This account supports Kerberos AES 256 bit encryption** option and click **OK** to apply the setting.


svc_zas Properties

User logon name: svc_zas @balasys.demo

User logon name (pre-Windows 2000): demo\ svc_zas

☐ Unlock account

Account options:

☐ Use only Kerberos DES encryption types for this account
☐ This account supports Kerberos AES 128 bit encryption.
☑ This account supports Kerberos AES 256 bit encryption.
☐ Do not require Kerberos preauthentication

Account expires
[✓ Never]
End of: 2022. február 12., szombat

Figure 15.38. setting encryption

Step 9. Install the Kerberos packages on the required server, for example on Zorp Authentication Server.

```
#:apt-get install krb5-user
```

Step 10. Provide the FQDN of the default realm during the installation process.

Step 11. Test Kerberos with the following commands. In the example the FQDN is BALASYS.DEMO.

```
#:kinit svc_zas@BALASYS.DEMO
#:klist -e
#:kdestroy
```

Step 12. Set Kerberos with the following commands:

```
#:ktutil
ktutil:addent -password -p svc_zas@BALASYS.DEMO -k 1 -e
aes256-cts-hmac-sha1-96
ktutil:addent -password -p svc_zas@BALASYS.DEMO -k 2 -e
aes256-cts-hmac-sha1-96
ktutil:addent -password -p svc_zas@BALASYS.DEMO -k 3 -e
aes256-cts-hmac-sha1-96
ktutil:addent -password -p svc_zas@BALASYS.DEMO -k 4 -e
```

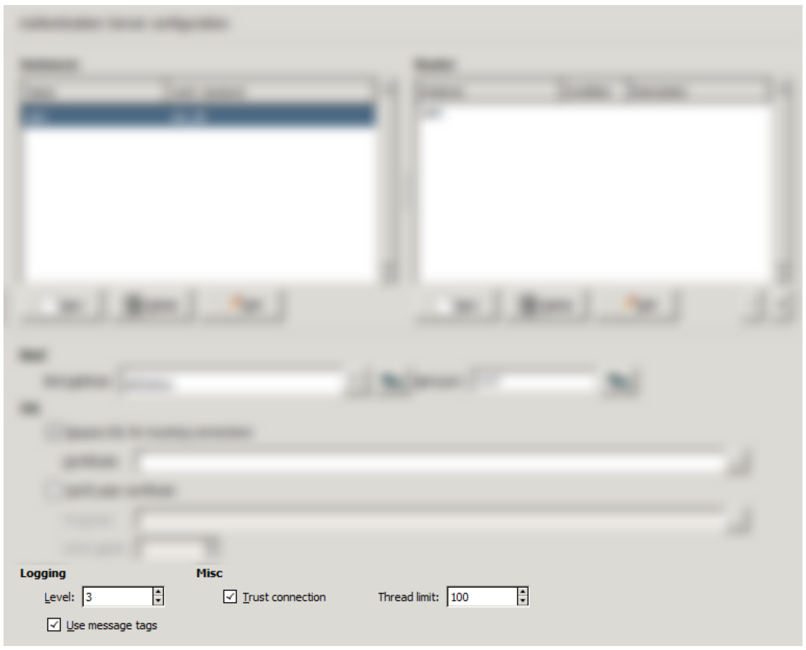
```
aes256-cts-hmac-sha1-96
ktutil:addent -password -p svc_zas@BALASYS.DEMO -k 5 -e
aes256-cts-hmac-sha1-96
ktutil:addent -password -p svc_zas@BALASYS.DEMO -k 6 -e
aes256-cts-hmac-sha1-96
ktutil:wkt /etc/krb5.keytab
ktutil:exit
#:chown zas /etc/krb5.keytab
```

15.3.5. Configuring the authentication agent

The Zorp Authentication Agent has to be installed on the client machines when outband authentication is used on the network. For detailed instructions about how to install the authentication agent, see *Chapter 6, Installing the Zorp Authentication Agent (ZAA)* in *Zorp Professional 7 Installation Guide* and *Zorp Authentication Agent Manual*.

15.4. Logging in ZAS

Logging in ZAS can be configured in the **Logging** section of the **Authentication server** ZMC component. The parameters related to logging in ZAS are the following:



Logging
Level: 3
☑ Use message tags

Misc
☑ Trust connection
Thread limit: 100

Figure 15.39. Configuring logging in ZAS

- **Log level**: It is the verbosity level of the logs. Level 3 is the default value, and is usually sufficient. Log level 0 does not produce log messages, while log level 10 logs every small event, and shall only be used for debugging purposes.
- **Trust connection**: This parameter permits password-based authentication methods even for unencrypted connections. The default value is: 0 (false).

If this parameter is ON, the password is accepted even if the connection between Zorp and ZAS is not based on Transport Layer Security (TLS), otherwise it is not.

- **Thread limit**: This value defines the number of threads ZAS can start. The graphical user interface now sets the ZAS default *Thread limit* to 100. It is possible though to set a different value for the *Thread limit*. Set this value according to the anticipated number of stacked connections. If the *Thread limit* is too low, the Zorp proxies stacking ZAS will experience delays and refused connection attempts. A suggested method to calculate this number, is to monitor the log for "Too many running threads, waiting for one to become free" line and to increase Zorp/ZAS/ZCV *Thread limit* parameter accordingly.
  Note, that if the *Thread limit* is already set with the ZAS "--threads=" option in the init script, then that option takes precedence. The value defined in the init script is cleared with each upgrade though, therefore this value can easily be updated at each upgrade to the preferred value, defined in the GUI.

Chapter 16. Virtual Private Networks

This chapter explains how to build encrypted connections between remote networks and hosts using Virtual Private Networks (VPNs).

16.1. Virtual Private Networking basics

Computers and even complete networks often have to be connected across the Internet, like in the case of organizations having multiple offices or employees doing remote work. In such situations it is essential to encrypt the communication to prevent anyone unauthorized from obtaining sensitive data. Virtual Private Networks (VPNs) solve the problem of communicating confidentially over an untrusted, public network.

VPNs retain the privacy, authenticity, and integrity of the connection, and ensure that the communication is not eavesdropped or modified. VPN traffic is transferred on top of standard protocols over regular networks (for example, the Internet) by encapsulating data and protocol information of the private network within the protocol data of the public network. As a result, nobody can recover the tunneled data by examining the traffic between the two endpoints.

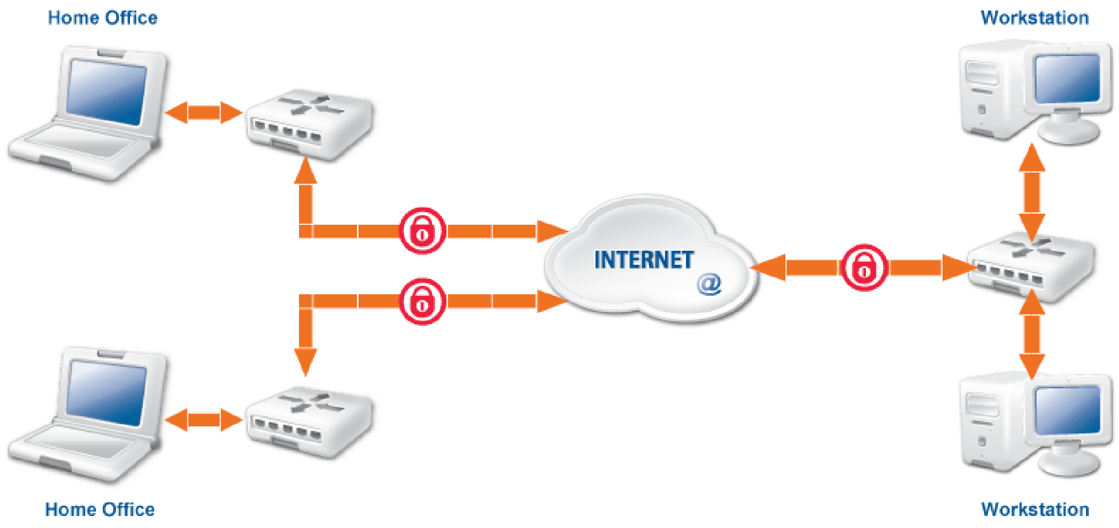


Figure 16.1. Virtual Private Networks

VPNs are commonly used in the following situations:

- to connect the internal networks of different offices of an organization
- to allow remotely-working employees access to the internal network
- to transfer unencrypted protocols in a secure, encrypted channel — without having to modify the original protocol
- to secure Wi-Fi networks

16.1.1. Types of VPN

Different VPN solutions use different methods to encrypt the communication. The main VPN types are IPSec, SSL/TLS, PPTP, and L2TP, with each type having many different implementations. Zorp supports the following VPN solutions:

- IPSec (strongSwan)
- SSL (OpenVPN)

16.1.2. VPN topologies

The topology of a VPN determines what is connected using the VPN. The basic VPN topologies are the following:

- *Peer-to-Peer*: It connects two hosts. (It is also called Point-to-Point VPN.)
- *Peer-to-Network*: It connects a single host to a network. This is the most common VPN topology, regularly used to allow remote workers access to the intranet of the organization. (it is also called Point-to-LAN VPN.)
- *Network-to-Network*: It completely connects two subnetworks. This solution is commonly used to connect the local networks of an organization having multiple offices. (it is also called LAN-to-LAN VPN.)

In every case, the VPN tunnel is created between two endpoints: the connecting hosts, or the firewall of the connecting network. The IP addresses of the connected networks or hosts can be fix (Fix IP connections) or dynamic (so called Roadwarrior connections). Roadwarrior connections are typically Peer-to-Network connections, where many peers (roadwarrior clients) can access the protected network.

16.1.3. The IPSec protocol

IP Security (IPSec) is a group of protocols that authenticate and encrypt every IP packet of a data stream. IPSec operates at the network layer of the OSI model (layer 3), so it can protect both TCP and UDP traffic. IPSec is also part of IPv6.

IPSec uses the Encapsulating Security Payload (ESP) and Authentication Header (AH) protocols to secure data packets, and the Internet Key Exchange (IKE) protocol to exchange cryptographic keys. IPSec has the following two modes:

- *Transport mode*: It is used to create peer-to-peer VPNs. Only the data part of the IP packet is encrypted, the header is not modified.
- *Tunnel mode*: It builds a complete IP tunnel to create Network-to-Network VPNs.

The IPSec implementation used by Zorp has two main components. Pluto is a userspace application responsible for the key exchange when building VPN connections. KLIPS is a kernel module that handles the encryption and transmission of the tunneled traffic after the VPN connection has been established.

16.1.4. The OpenVPN protocol

OpenVPN creates a VPN between the endpoints using an SSL/TLS channel. OpenVPN operates at the TCP layer of the OSI model (layer 4). The SSL channel is usually created using UDP packets, though it is possible to use TCP. Using SSL enables the endpoints to authenticate each other using certificates.

The OpenVPN server can 'push' certain parameters to the clients, for example, IP addresses, routing commands, and other connection parameters. OpenVPN transfers all communication using a single IP port.

The connecting clients receive an internal IP address, similarly to DHCP. This IP address is valid only within the VPN tunnel, and usually belongs to a virtual subnet.

OpenVPN creates VPN tunnels between virtual interfaces. These interfaces have internal IP addresses that are independent from the IP addresses of the physical interfaces, and are visible only from the VPN tunnels.

OpenVPN runs completely in userspace; the user does not need special privileges to use it. The kernel running on the host must support the virtual interfaces used to create the VPN tunnels.

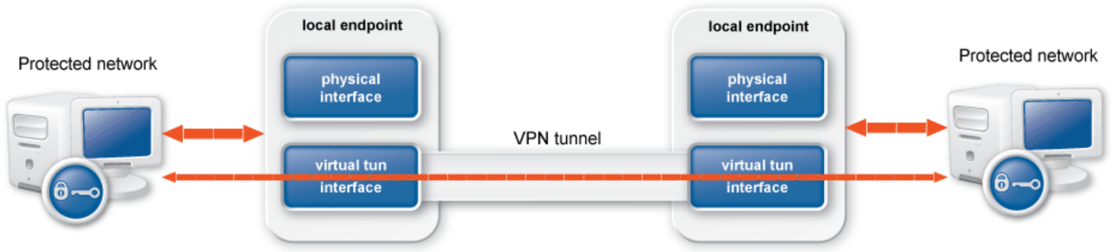
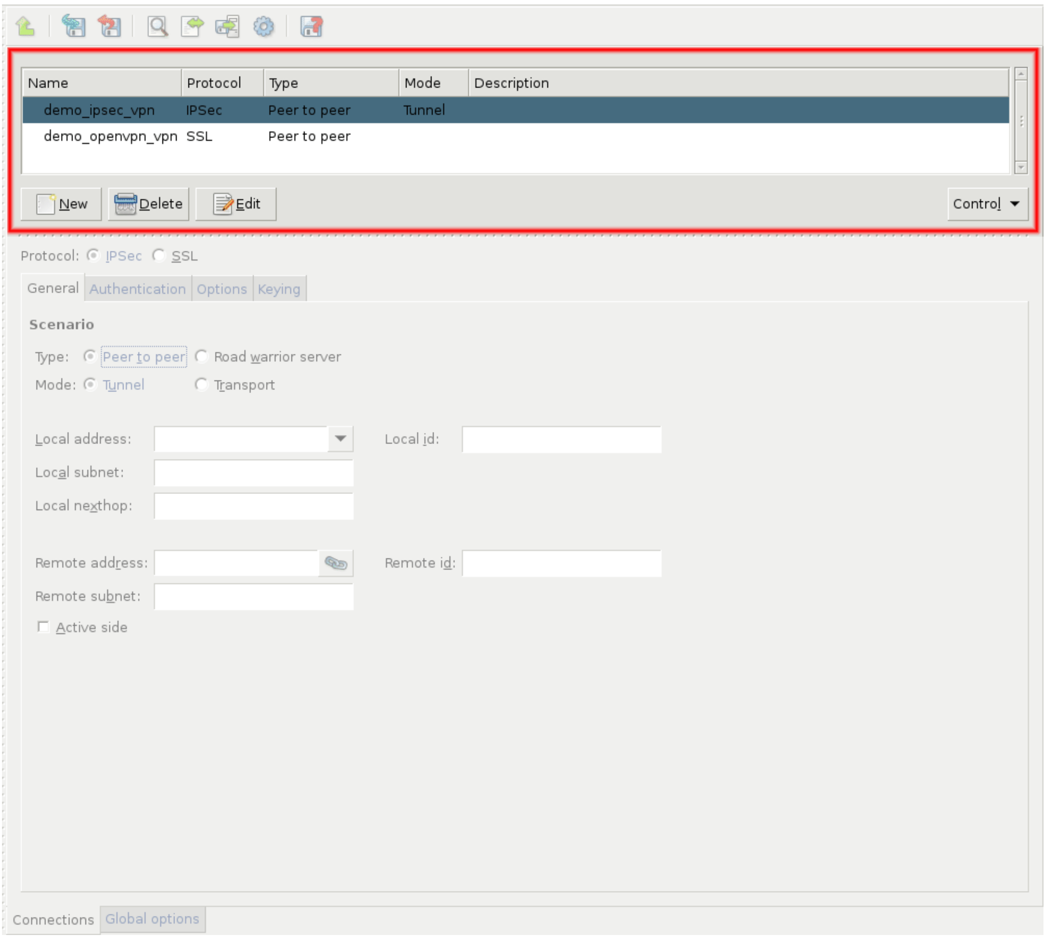


Figure 16.2. The operation of OpenVPN

16.2. Using VPN connections

VPN connections can be configured using the **VPN** ZMC component. Before starting to configure VPN connections, add this component to the host (see *Procedure 3.2.1.3.1, Adding new configuration components to host (p. 22)* for details).

Note
If only IPSec traffic is required to be forwarded without terminating the tunnel on Zorp, see *Procedure 16.3.4, Forwarding IPSec traffic on the packet level (p. 441)*.



Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Protocol: [✓ IPSec] SSL

Scenario

Type: [✓ Peer to peer] Road warrior server

Mode: [✓ Tunnel] Transport

Figure 16.3. Using VPN connections

Use the **New**, **Delete**, and **Edit** buttons to create, remove, or rename VPN connections. Clicking on **Control** displays a drop-down menu to start, stop, or restart the selected connections.

The VPN ZMC component automatically creates the required *`ipsec`* and *`tun`* interfaces for the configured VPN tunnels. Use these interfaces to define Zorp services that can be accessed through the VPN tunnel. Firewall rules can use these interfaces like a regular, physical network interface. The general procedure of using VPNs is as follows:

16.2.1. Procedure – Using VPN connections

Step 1. Create the certificates required for authentication using the Zorp PKI. See *Procedure 11.3.8.2, Creating certificates (p. 276)* for details.

Step 2. Configure the VPN tunnel using the VPN ZMC component. See *Section 16.3, Configuring IPSec connections (p. 430)* and *Section 16.4, Configuring SSL (OpenVPN) connections (p. 441)* for details.

Tip
To create Peer to Peer or Network to Network connections, use IPSec.

To create Roadwarrior servers, use SSL.

Step 3. Create services that can be accessed from the VPN tunnel using the Zorp ZMC component.

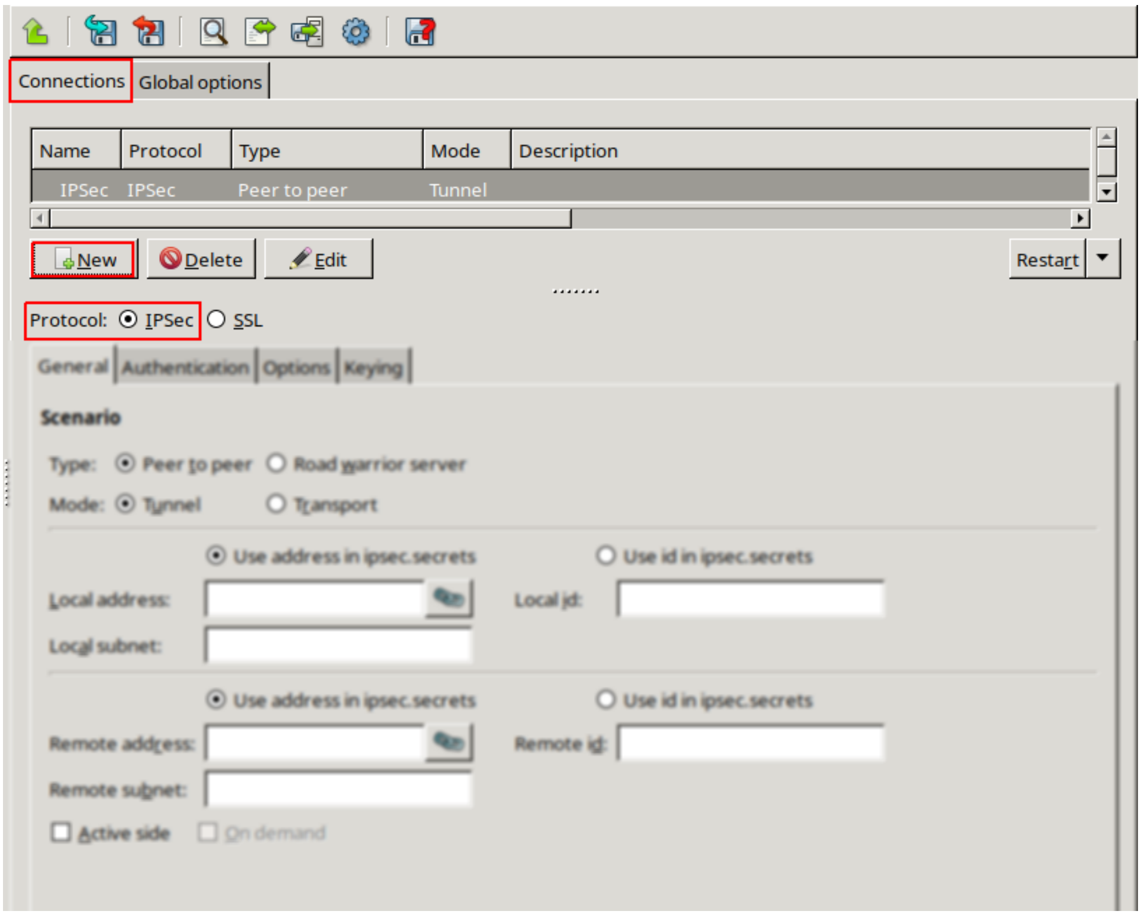
Step 4. Configure the remote endpoints (for example, roadwarrior clients) that will access the VPN tunnel. This process may involve installing VPN client software and certificates, and so on.

16.3. Configuring IPSec connections

This section explains how to configure IPSec VPN connections.

16.3.1. Procedure – Configuring IPSec connections

Step 1. Navigate to the VPN component of the Zorp host that will be the endpoint of the VPN connection. Select the **Connections** tab.



Name	Protocol	Type	Mode	Description
IPSec	IPSec	Peer to peer	Tunnel	

Protocol: [✓ IPSec] SSL

Type: [✓ Peer to peer] Road warrior server

Mode: [✓ Tunnel] Transport

[✓ Use address in ipsec.secrets] Use id in ipsec.secrets

[✓ Use address in ipsec.secrets] Use id in ipsec.secrets

Figure 16.4. Configuring IPSec connections

Step 2. Click **New** and enter a name for the connection.

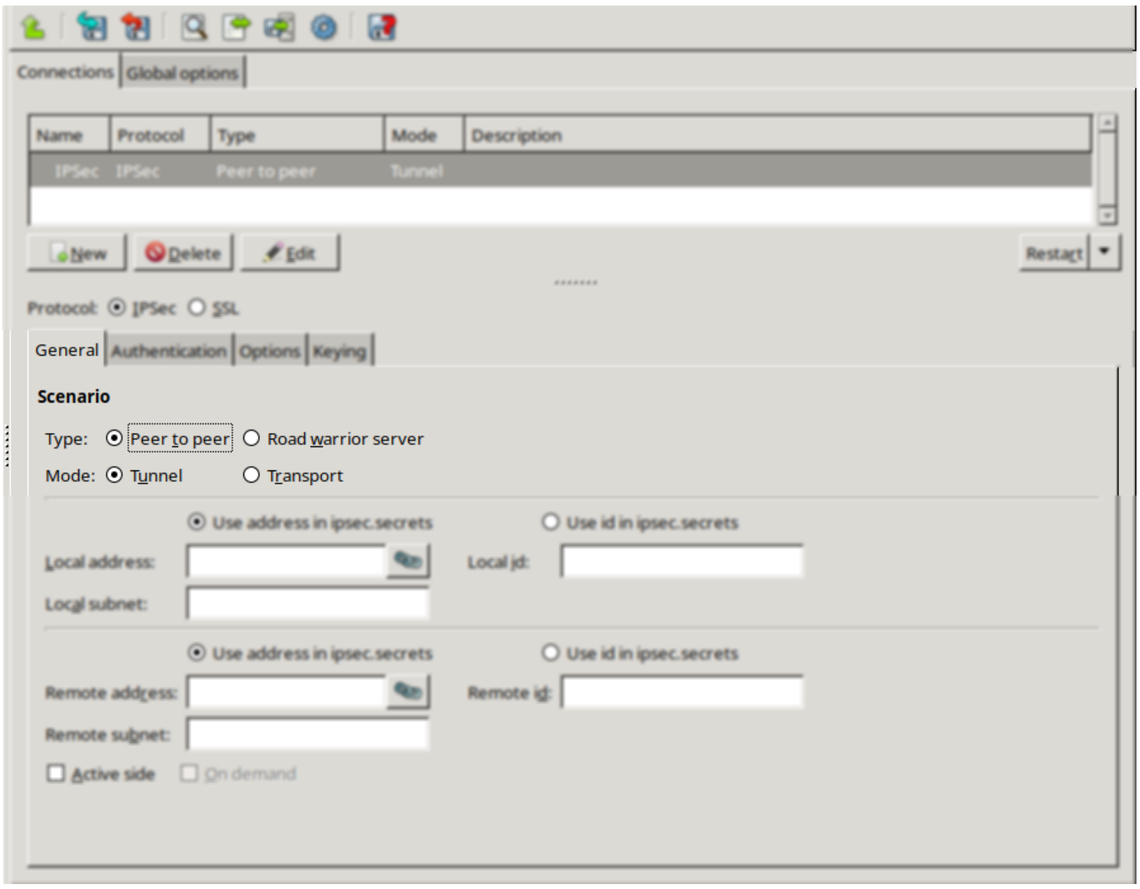
Step 3. Select the **IPSec** protocol option.

Step 4. Set the VPN topology and the transport mode in the **Scenario** section on the **General** tab.

- To create a Peer-to-Peer connection, select the **Peer to Peer** and the **Transport** options.
- To create a Peer-to-Network connection, select the **Peer to Peer** and the **Tunnel** options.

- To create a Roadwarrior server, select the **Roadwarrior server** and the **Transport** options.
- To create a Network-to-Network connection, select the **Peer to Peer** and the **Tunnel** options.

Note
When creating a Network-to-Network connection, the two endpoints of the VPN tunnel do NOT use the VPN to communicate with each other. To encrypt the communication of the endpoints, create a separate Peer-to-Peer connection.



Name	Protocol	Type	Mode	Description
IPSec	IPSec	Peer to peer	Tunnel	

Protocol: [✓ IPSec] SSL

Type: [✓ Peer to peer] Road warrior server

Mode: [✓ Tunnel] Transport

[✓ Use address in ipsec.secrets] Use id in ipsec.secrets

Local address:
Local id:
Local subnet:

[✓ Use address in ipsec.secrets] Use id in ipsec.secrets

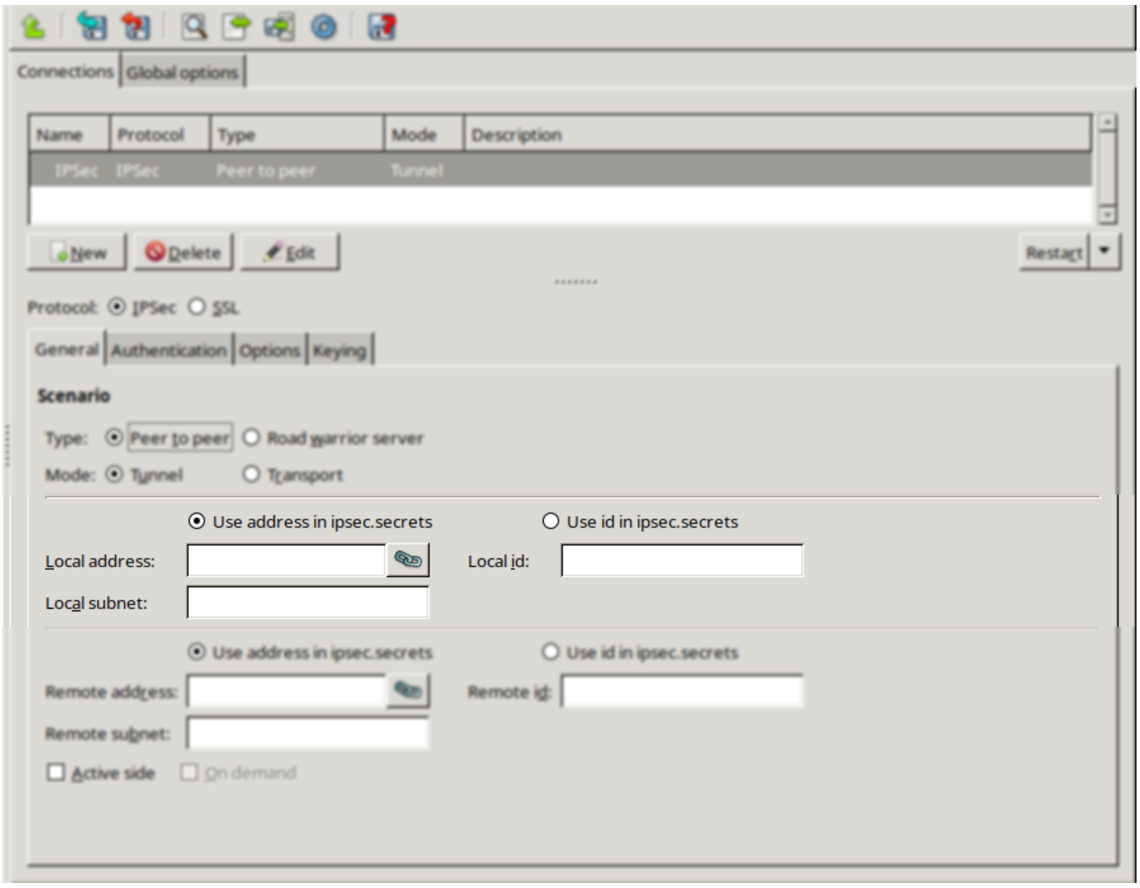
Remote address:
Remote id:
Remote subnet:

☐ Active side ☐ On demand

Figure 16.5. Selecting the IPSec scenario

Step 5. Configure the local networking parameters.
These parameters affect the Zorp endpoint of the VPN connection. Set the following parameters:

- **Local address**: Select the IP address that Zorp will use for the VPN connection.
- **Local ID**: It is the ID of the Zorp endpoint in the VPN connection. Leave this field blank unless there are difficulties in establishing the connection with the remote VPN application. If the **Local ID** is set, the **Use ID in ipsec.secrets** option might also need to be set.
- **Local subnet**: It is the subnet behind Zorp that will be accessible using the VPN tunnel. This option is available only for Peer-to-Network and Network-to-Network connections.



Name	Protocol	Type	Mode	Description
IPSec	IPSec	Peer to peer	Tunnel	

Protocol: [✓ IPSec] SSL

Scenario

Type: [✓ Peer to peer] Road warrior server

Mode: [✓ Tunnel] Transport

[✓ Use address in ipsec.secrets] Use id in ipsec.secrets

Local address:

Local id:

Local subnet:

[✓ Use address in ipsec.secrets] Use id in ipsec.secrets

Remote address:

Remote id:

Remote subnet:

Active side

On demand

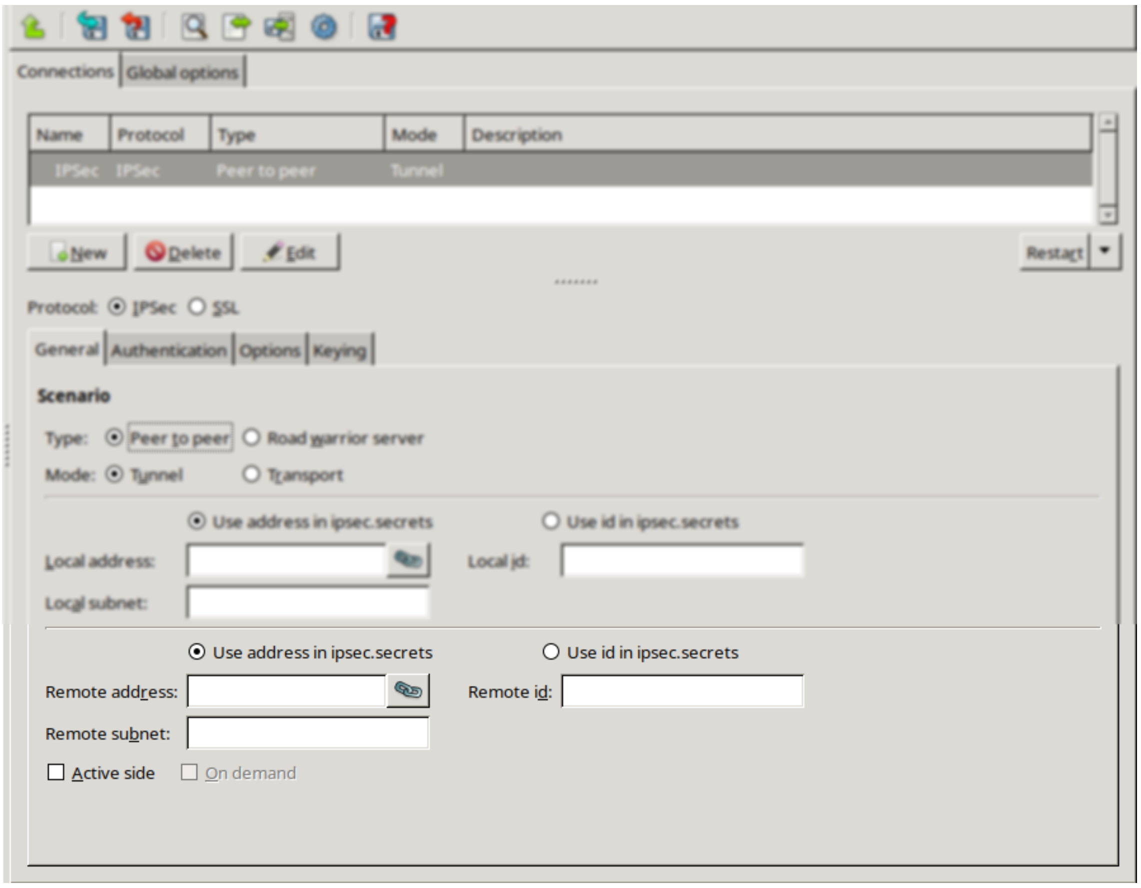
Figure 16.6. Configuring local networking parameters

Step 6. Configure the networking parameters of the remote endpoint. Set the following parameters:

- **Remote address**: It is the IP address of the remote endpoint. It does not apply for roadwarrior VPNs.
- **Remote ID**: It is the ID of the remote endpoint in the VPN connection. Leave this field blank unless there are difficulties in establishing the connection with the remote VPN application. If the **Remote ID** is set the **Use ID in ipsec.secrets** option might also need to be set.
- **Remote subnet**: It is the subnet behind the remote endpoint that will be accessible using the VPN tunnel. This option is available only for Peer-to-Network and Network-to-Network connections.

Note
Network-to-Network connections connect the subnets specified in the **Local subnet** and **Remote subnet** parameters.

Do not specify the subnet parameter for the peer side of Peer-to-Network connections, leave either the **Local subnet** or the **Remote subnet** parameter empty.



Name	Protocol	Type	Mode	Description
IPSec	IPSec	Peer to peer	Tunnel	

Protocol: [✓ IPSec] SSL

Scenario

Type: [✓ Peer to peer] Road warrior server

Mode: [✓ Tunnel] Transport

[✓ Use address in ipsec.secrets] Use id in ipsec.secrets

Local address: Local id:

Local subnet:

[✓ Use address in ipsec.secrets] Use id in ipsec.secrets

Remote address: Remote id:

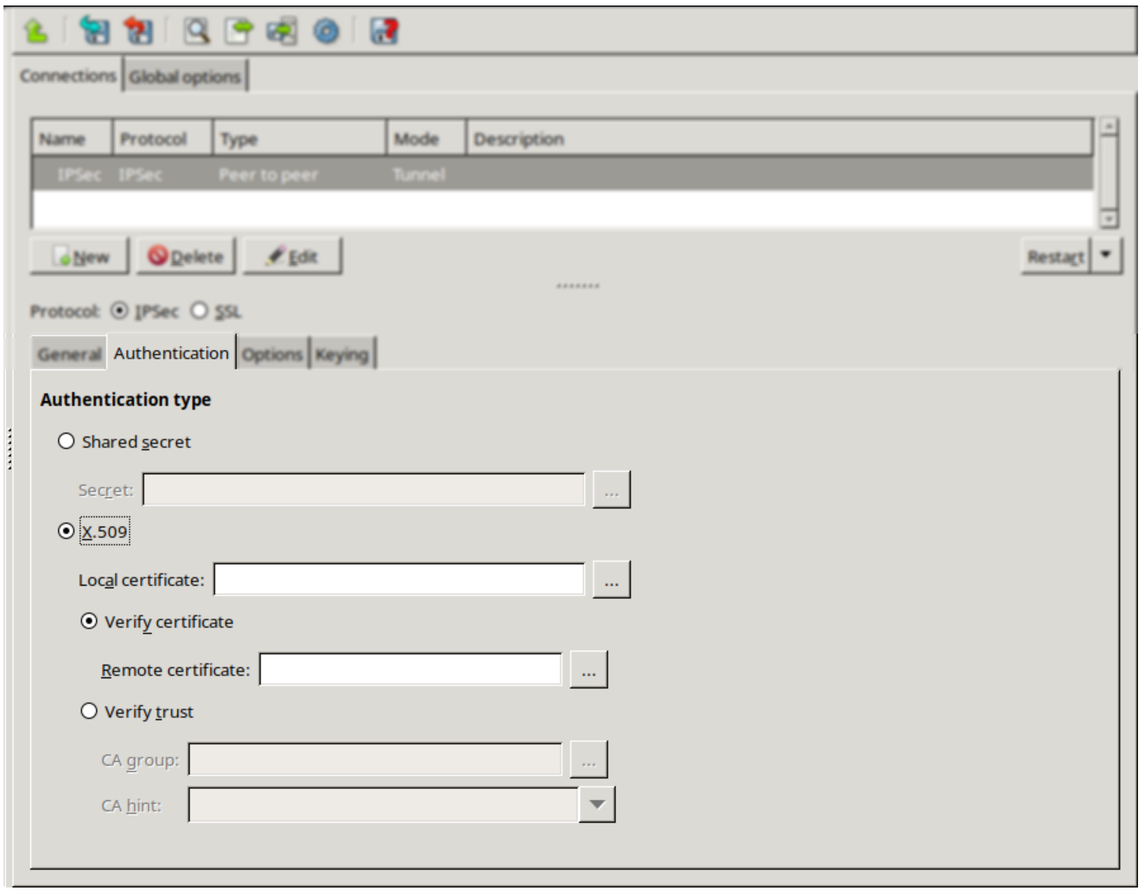
Remote subnet:

☐ Active side ☐ On demand

Figure 16.7. Configuring remote networking parameters

Step 7. When configuring **Peer-to-Peer** or **Network-to-Network** connections, it is crucial that the endpoint operators cooperate. If the **Active side** option is selected, Zorp opens the VPN connection to the remote endpoint. It is possible to enable the **Active side** option on both sides, but if the tunnel is unstable, it is recommended to enable it only on one side.

Step 8. Click on the **Authentication** tab and configure authentication.



Name	Protocol	Type	Mode	Description
IPSec	IPSec	Peer to peer	Tunnel	

Protocol: [✓ IPSec] SSL

Authentication type

Shared secret

Secret:

[✓ X.509]

Local certificate:

[✓ Verify certificate]

Remote certificate:

Verify trust

CA group:

CA hint:

Figure 16.8. Configuring authentication

To use password-based authentication, select the **Shared secret** option and enter the password in the **Secret** field.

Note
Authentication using a shared secret is not a secure authentication method. Use it only if the remote endpoint does not support certificate-based authentication. Always use long and complicated shared secrets: at least twelve characters containing a mix of alphanumerical and special characters. Remember to change the shared secret regularly.

To use certificate-based authentication, select the **X.509** option and set the following parameters:

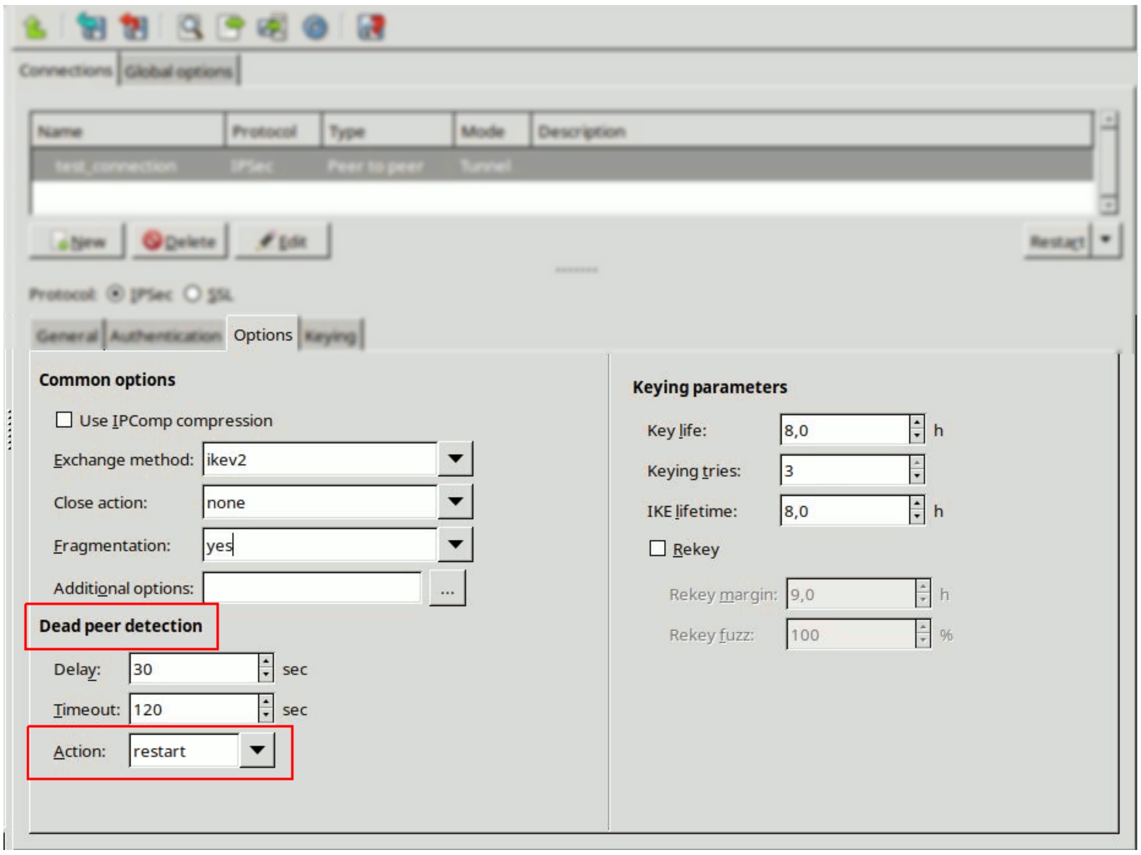
- **Local certificate**: Select a certificate available on the Zorp host. Zorp will show this certificate to the remote endpoint.
- If the remote endpoint has a specific certificate, select the **Verify certificate** option and select the certificate from the **Remote certificate** field. Zorp will use this certificate to verify the certificate of the remote endpoint.
- If there are several remote endpoints that can connect to the VPN tunnel, select the **Verify trust** option and select the trusted Certificate Authority (CA) group containing the CA certificate of the CA that issued the certificates of the remote endpoints from the **CA group**

field. Zorp will use this trusted CA group to verify the certificates of the remote endpoints. (See *Section 11.3.7, Trusted CAs (p. 267)* for details.)
Zorp sends the common name of the accepted CAs to the remote endpoint, so the client knows what kind of certificate is required for the authentication. Select a specific CA certificate using the **CA hint** option if only certificates signed by the selected CA are required to be accepted.

Note
See *Chapter 11, Key and certificate management in Zorp (p. 249)* for details on creating and importing certificates, CAs, and trusted CA groups required for certificate-based authentication.

Step 9. Before setting the action status of the Dead Peer Detection option, it is necessary that the two endpoint operators agree on the preferred settings. If earlier the **Active side** option was selected for Zorp, it is recommended to select the *`restart`* option of Action parameter. This way Zorp attempts to restart the VPN connection if the remote endpoint becomes unavailable.
If Zorp is on the passive side and earlier the **Active side** option was not enabled, it is recommended to set the Action parameter of the Dead Peer Detection to *`hold`* for Zorp and set this parameter to *`restart`* on the remote endpoint.

Note
Dead Peer Detection is effective only if enabled on both endpoints of the VPN connection. If Dead Peer Detection is enabled only on one side, and it is disabled on the other side it may lead to unreliable VPN connection. If Dead Peer Detection is not required, it must be disabled at both endpoints.


Connections | Global options

Name	Protocol	Type	Mode	Description
test_connection	IPSec	Peer to peer	Tunnel	

New | Delete | Edit | Restart

Protocol: [✓ IPSec] SSL

General | Authentication | Options | Keying

Common options

- [ ] Use IPComp compression
- Exchange method: ikev2
- Close action: none
- Fragmentation: yes
- Additional options:

Dead peer detection

- Delay: 30 sec
- Timeout: 120 sec
- Action: restart

Keying parameters

- Key life: 8,0 h
- Keying tries: 3
- IKE lifetime: 8,0 h
- [ ] Rekey
- Rekey margin: 9,0 h
- Rekey fuzz: 100 %

Figure 16.9. Configuring IPSec options

The following additional parameters can be configured for *Dead Peer Detection*:

- Delay
  This parameter defines the time interval in which informal messages are sent to the peer.

- Timeout
  This parameter defines the timeout interval after which all connections to a peer are deleted in case of inactivity.

- Action
  This parameter controls the usage of Dead Peer Detection protocol, where informal messages are periodically sent to check whether the connection toward the IPSec peer is live or not.

  The available values are: *clear*, *restart* and *none*.

  The values *clear, hold and restart activate Dead Peer Detection and instruct on the action to be taken in case of timeout.*

  If the parameter is set to *clear*, the connection shall be closed without any further action taken.

  If the parameter is set to *hold*, matching traffic will be searched for and renegotiation on the connection will be tried.

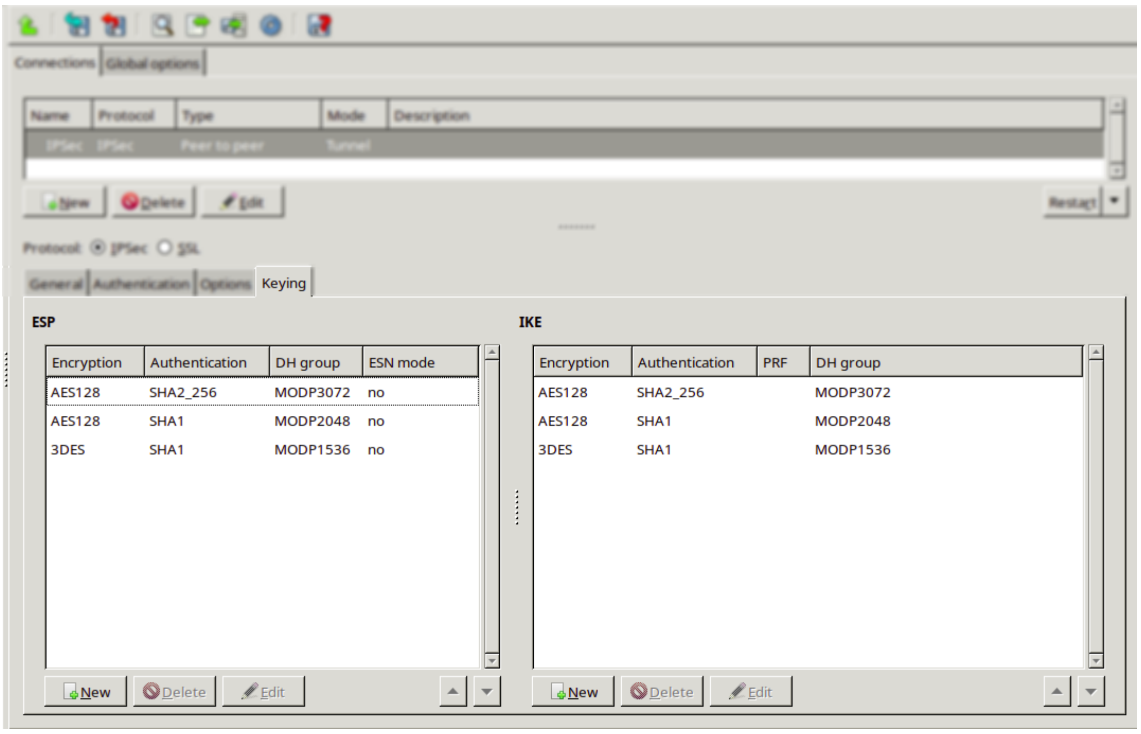
If the parameter is set to *restart*, an immediate attempt will take place for renegotiating the connection.

If the parameter is set to *none*, no more Dead Peer Detection messages will be sent to the peer.

Step 10. Set other options if needed. See *Section 16.3.2, IPSec options (p. 438)* for details.

Note
By default, Zorp 3 F5 and later versions use the IKEv2 key exchange protocol. However, earlier versions support only the IKEv1 protocol. Change the **Options > Exchange protocol** option to IKEv1 when the remote endpoint of the VPN connection is running Zorp 3.4 LTS or earlier.

Step 11. Configure the parameters of the Keying tab, if necessary.



Name	Protocol	Type	Mode	Description
IPSec	IPSec	Peer to peer	Tunnel	

ESP

Encryption	Authentication	DH group	ESN mode
AES128	SHA2_256	MODP3072	no
AES128	SHA1	MODP2048	no
3DES	SHA1	MODP1536	no

IKE

Encryption	Authentication	PRF	DH group
AES128	SHA2_256		MODP3072
AES128	SHA1		MODP2048
3DES	SHA1		MODP1536

Figure 16.10. Keying tab parameters

- Encapsulating Security Payload (ESP)
  This list presents the Encapsulating Security Payload (ESP) encryption and authentication algorithms that shall be used for the actual connection.

  If the DH group is also specified, it defines that *Diffe-Hellman (DH) exchange shall be included in re-keying or in initial negotiation.*

  The ESN parameter defines whether Extended Sequence Number (ESN) support with the peer is enabled or not. The default value is 'no'.

- Internet Key Exchange (IKE)
  This list presents the Internet Key Exchange (IKE) encryption and authentication algorithms that shall be used for the actual connection.

  If the DH group is also specified, it defines that *Diffe-Hellman exchange shall be included in re-keying or in initial negotiation.*

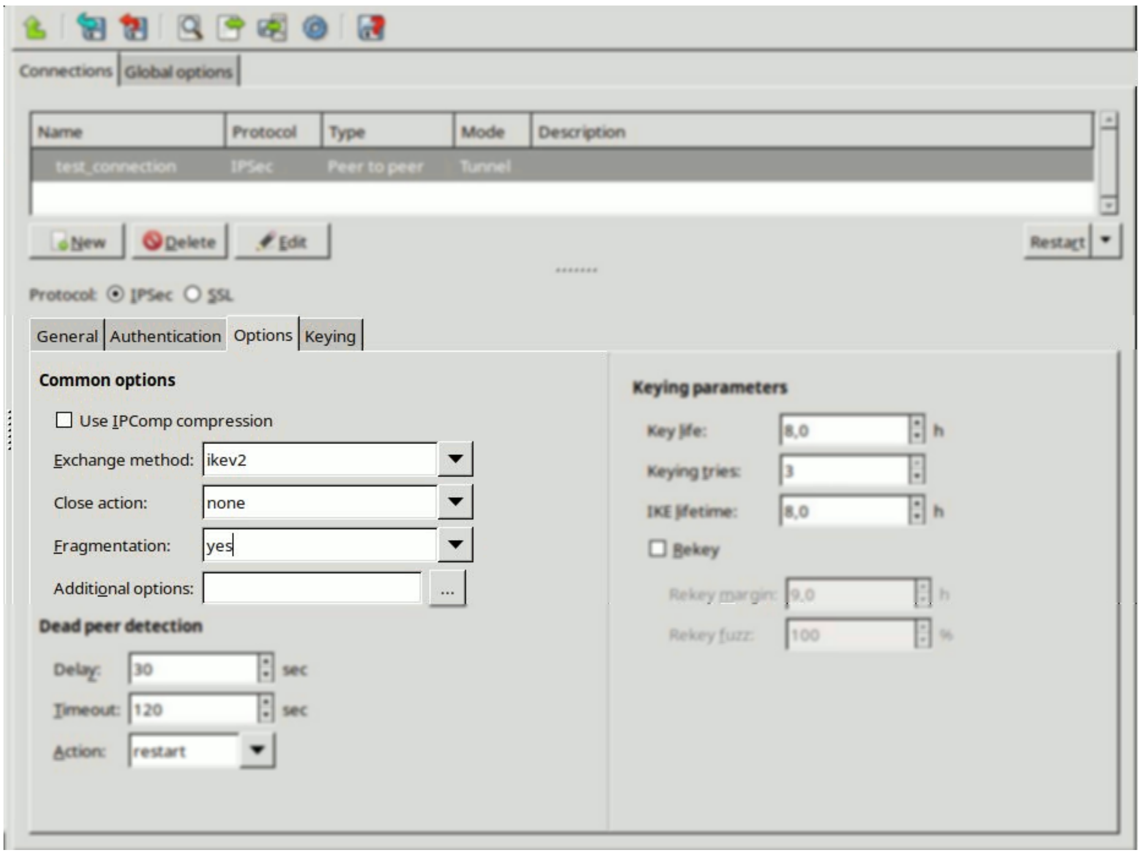
  If no *Pseudo Random Function (PRF)* algorithm is configured, the algorithms defined for integrity are proposed as PRF.

16.3.2. IPSec options

Global parameters that apply to every IPSec VPN connection of the Zorp host can be set on the **Global options** tab.

Set special options of a particular IPSec VPN connection on the **Connections** tab and the **Options** and **Keying** submenu tabs.

Besides the Dead Peer detection parameters, as introduced in *Section 16.3, Configuring IPSec connections (p. 430)*, there are additional parameters that can be configured Under the *Options* tab.



Name	Protocol	Type	Mode	Description
test_connection	IPSec	Peer to peer	Tunnel	

Protocol: [✓ IPSec] SSL

Common options

☐ Use IPComp compression

Exchange method: ikev2

Close action: none

Fragmentation: yes

Additional options:

Dead peer detection

Delay: 30 sec

Timeout: 120 sec

Action: restart

Keying parameters

Key life: 8,0 h

Keying tries: 3

IKE lifetime: 8,0 h

☐ Rekey

Rekey margin: 9,0 h

Rekey fuzz: 100 %

Figure 16.11. Configuring 'Common options' parameters at IPSec Options tab

Common options

- *Use IPComp compression*
  If this parameter is enabled, that is the parameter is checked in, the daemon accepts compressed and uncompressed data as well. If the parameter is not enabled, that is, the parameter is not checked in, the daemon accepts only uncompressed data.

- Exchange method
  The available values are: *ikev1*, *ikev2*.

  The key exchange method can be selected here for initializing the connection, that is *ikev1* or *ikev2*.

- Close action
  The available values are: *none*, *clear*, *hold* and *restart*.

  This parameter defines the action to take if the remote peer unexpectedly closes. This parameter is not supported for ikev1 connections.

- Fragmentation
  The available values are: *yes*, *accept*, *force* and *no*.

  This parameter enables Internet Key Excahnge (IKE) fragmentation. Note that fragmented messages arriving from a peer are always processed, regardless of this parameter option.

  If this parameter is set to *yes*, that is, checked in, and the peer supports it, any oversized IKE message will be fragmented.

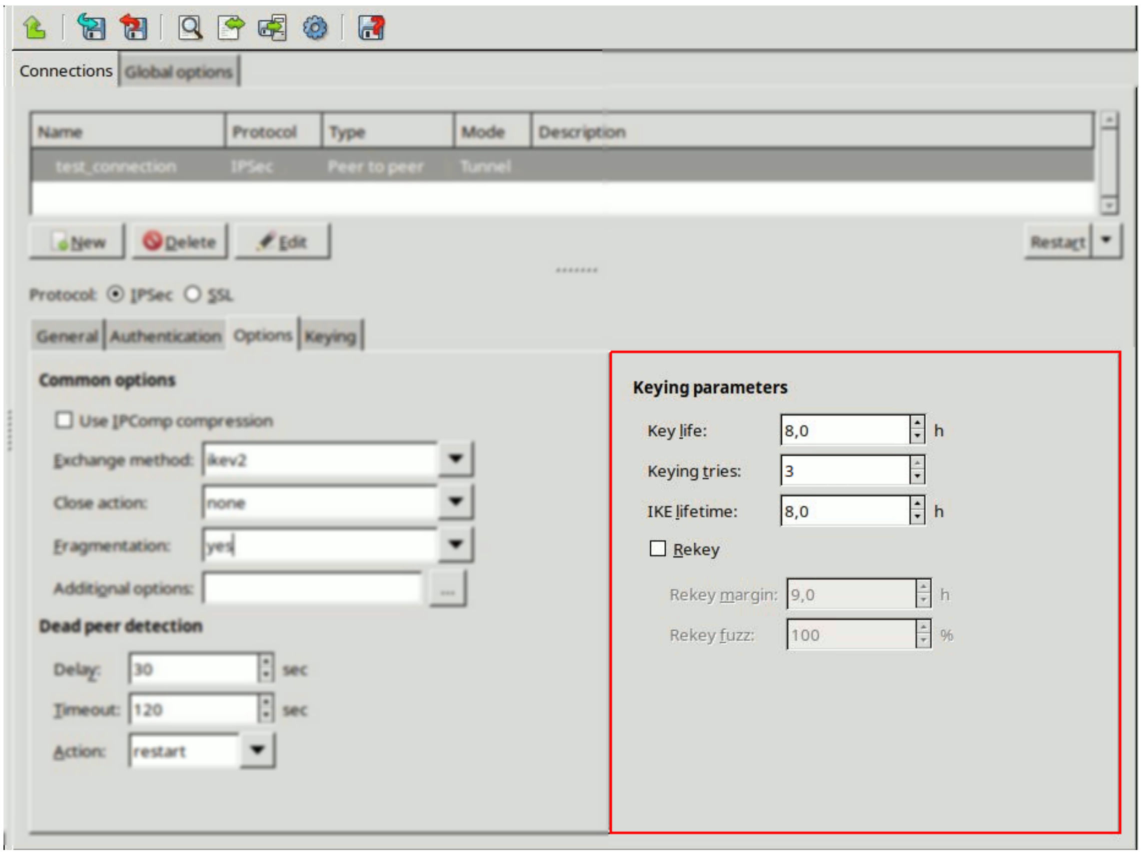
  If the parameter is set to *accept* fragmented content is supported arriving from the peer, yet the daemon does not send fragmented messages.

  If the parameter is set to *force* the initial IKE message will be fragmented.

- Additional options
  This parameter enables the user to provide any additional StrongSwan parameter manually that is not available in the GUI. Also, if the configuration of a parameter is available though in the GUI, but the required setting is not, it can be manually defined here. Even already defined parameter configuration settings can be overwritten at **Additional options**, as the configuration will use the latest definition of the parameter. The parameters have to be provided in the format described on the StrongSwan documentation site, available at:

  https://wiki.strongswan.org/projects/strongswan/wiki/ConnSection.

The **Keying parameters** section of the **Options** tab specifies key-handling and key-exchange parameters. Modify these parameters only if it is necessary for compatibility with the remote endpoint.



Connections | Global options

Name	Protocol	Type	Mode	Description
test_connection	IPSec	Peer to peer	Tunnel	

Protocol: [✓ IPSec] SSL

Common options

☐ Use IPComp compression

Exchange method:	ikev2
Close action:	none
Fragmentation:	yes
Additional options:	

Dead peer detection

Delay:	30	sec
Timeout:	120	sec
Action:	restart	

Keying parameters

Key life:	8,0	h
Keying tries:	3	
IKE lifetime:	8,0	h

☐ Rekey

Rekey margin:	9,0	h
Rekey fuzz:	100	%

Figure 16.12. Configuring keying parameters at IPSec Options tab

Note
Do not modify these options unless it is required and it is perfectly clear how these parameter settings affect the configuration.

The options of the **Keying** tab specify the encryption used in the connection.

Keying parameters

- Key life
  This parameter defines how long the key connection shall last, from the negotiation until expiry.
- Key tries
  This parameter defines the number of attempts for negotiating or renegotiating the connection.
- IKE lifetime
  This parameter defines the length of the keying channel connection before it is renegotiated.
- Rekey
  Enabling this parameter requires the connection to be renegotiated when it is about to expire.

Disabling this parameter will result in the daemon not requesting renegotiation, nevertheless, it does not prevent from responding to renegotiation requested from the other end.

16.3.3. Global IPSec options

The following options apply to every IPSec VPN tunnel. These settings are available on the **Global options** tab.

- **Verbose IKE**: Include log messages of the Internet Key Exchange (IKE) protocol in the logs.
- **Cache CRLs**: This parameter can be set to **ON**, that is `cachecrls=yes`, or to **OFF**, that is `cachecrls=no`. If Certificate Revocation List (CRL) caching is enabled, local caching of CRLs is activated and no new CRL is picked up until the locally cached CRL has expired. The cached CRL is stored in */etc/ipsec.d/crls* under a unique filename. As soon as it has expired, it is replaced with an updated CRL.
- **Strict CRL policy**: The CRL handling policy is quite tolerant by default, that is, the `strictcrlpolicy` is set to **no** by default. Consequently, in case a CRL is expired, only a warning is issued and another peer CRL is automatically accepted. If a more strict CRL policy is required, this parameter has to be enabled here, the `strictcrlpolicy` parameter will be set to **yes**. If the parameter `strictcrlpolicy` is enabled, no certificate will be accepted from a peer until no corresponding CRL is present in */etc/ipsec.conf*. If this parameter is enabled it is crucial therefore to make sure that the CRLs are updated in time.

For details on the other options, see the *strongSwan* documentation available at *http://wiki.strongswan.org/*.

16.3.4. Procedure – Forwarding IPSec traffic on the packet level

If IPSec traffic on Zorp is not required to be terminated, only to be forwarded, create packet filtering rules for the Encapsulating Security Payload (ESP) (protocol number 50) and AH (protocol number 51) protocols. Complete the following steps:

Step 1. Select the **Packet filter** ZMC component from the configuration tree, and click on the **Ruleset** tab.

Step 2. In the **Hierarchy** column, open the **filter** table, and select the **FORWARD** chain.

Step 3. Click **New Child**, enter `50` into the **Protocol** field, and click **OK**. Optionally, also specify the source and destination interfaces.

Step 4. Select the **FORWARD** chain, click **New Child**, enter `51` into the **Protocol** field, and click **OK**.

Step 5. Click **Generate ruleset**.

Step 6. Commit and upload the configuration changes and reload the Packet filter component.

16.4. Configuring SSL (OpenVPN) connections

This section explains how to configure SSL VPN connections.

16.4.1. Prerequisities for configuring SSL (OpenVPN) connections

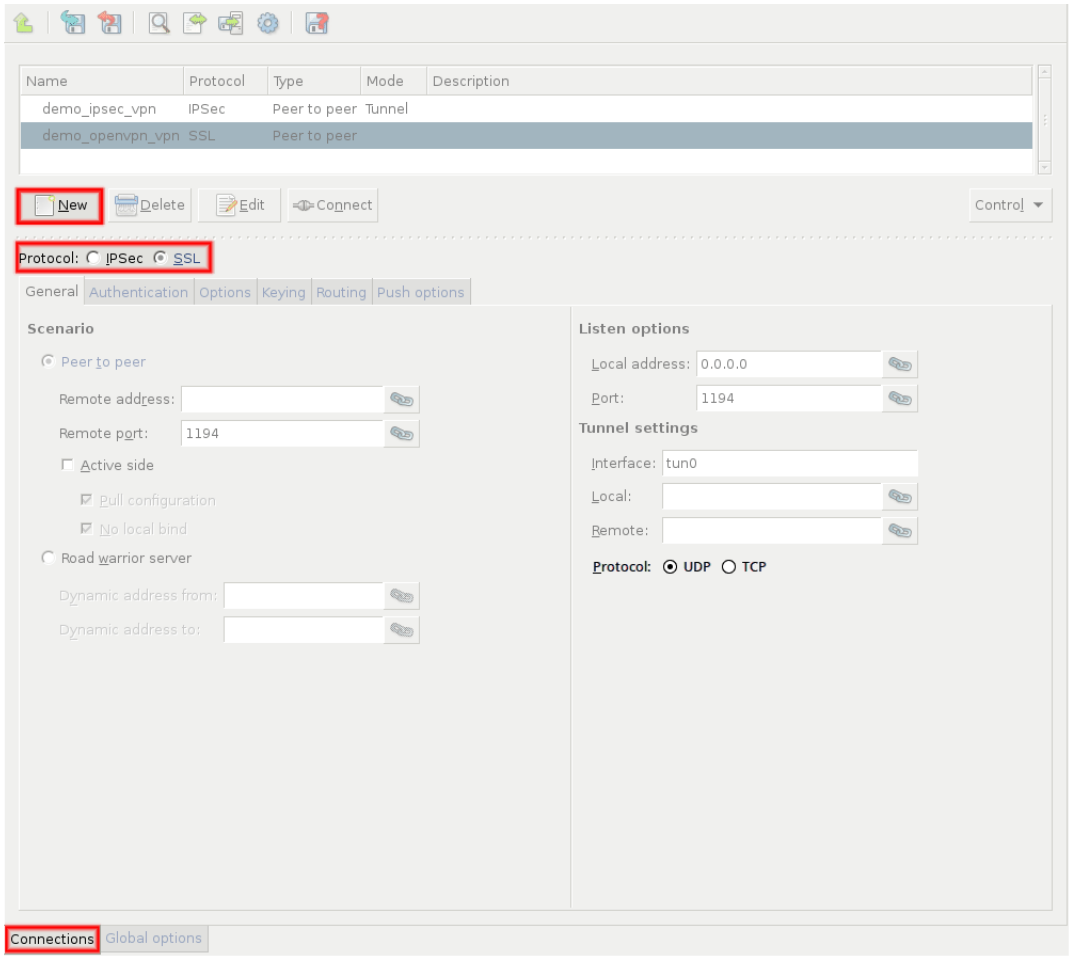
- Install the OpenVPN package on the Host, if it is not yet installed:

```
apt install openvpn
```

- Make sure that the VPN ZMC component is added to the host. For details on adding the VPN component, see *Procedure 3.2.1.3.1, Adding new configuration components to host (p. 22)*.

16.4.2. Procedure – Configuring SSL connections

Step 1. Navigate to the VPN component of the Zorp host that will be the endpoint of the VPN connection. Select the **Connections** tab.



Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Protocol: IPSec [✓ SSL]

Scenario

[✓ Peer to peer]

Remote address:

Remote port: 1194

☐ Active side

☑ Pull configuration

☑ No local bind

Road warrior server

Dynamic address from:

Dynamic address to:

Listen options

Local address: 0.0.0.0

Port: 1194

Tunnel settings

Interface: tun0

Local:

Remote:

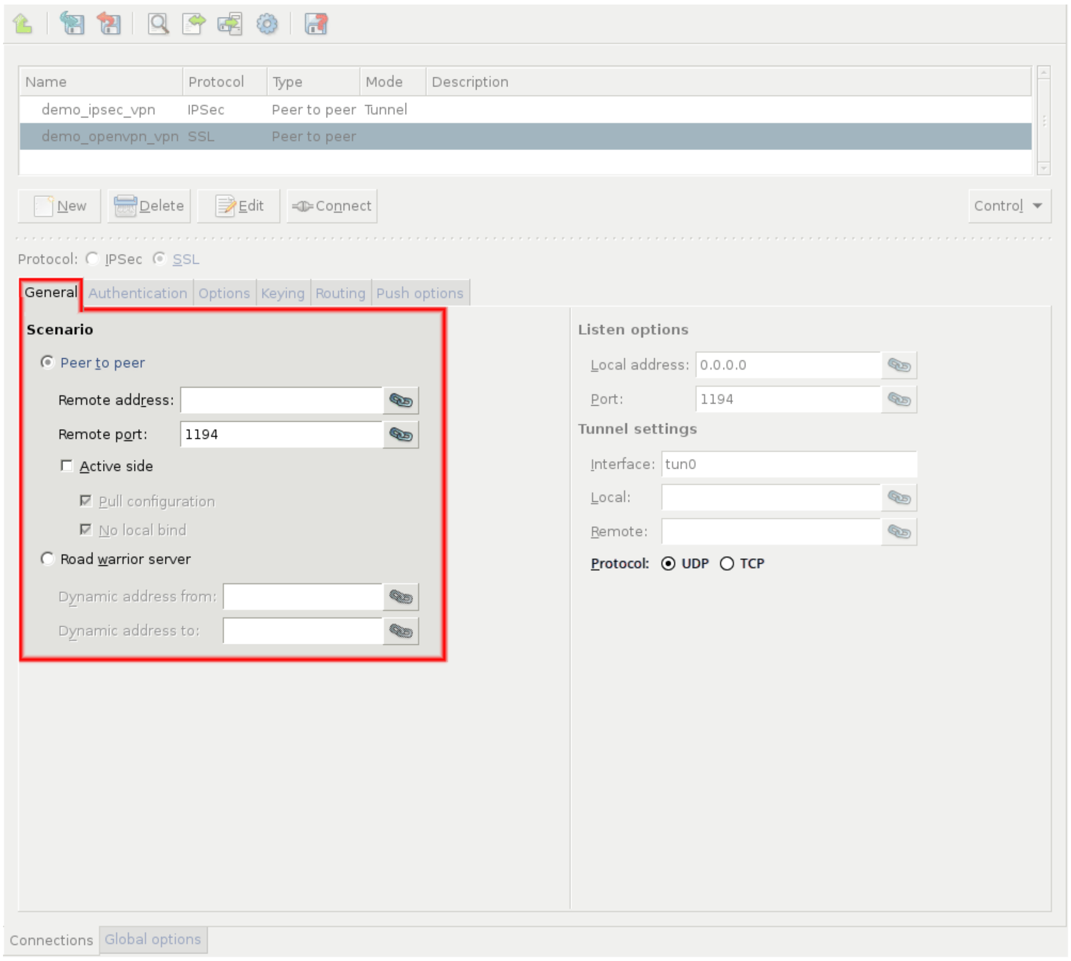
Protocol: [✓ UDP] TCP

Figure 16.13. Configuring SSL (OpenVPN) connections

Step 2. Click **New** and enter a name for the connection.

Step 3. Select the **SSL** protocol option.

Step 4. Set the VPN topology in the **Scenario** section.



Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Protocol: IPSec [✓ SSL]

Scenario

[✓ Peer to peer]

Remote address:

Remote port: 1194

☐ Active side

☑ Pull configuration

☑ No local bind

Road warrior server

Dynamic address from:

Dynamic address to:

Listen options

Local address: 0.0.0.0

Port: 1194

Tunnel settings

Interface: tun0

Local:

Remote:

Protocol: [✓ UDP] TCP

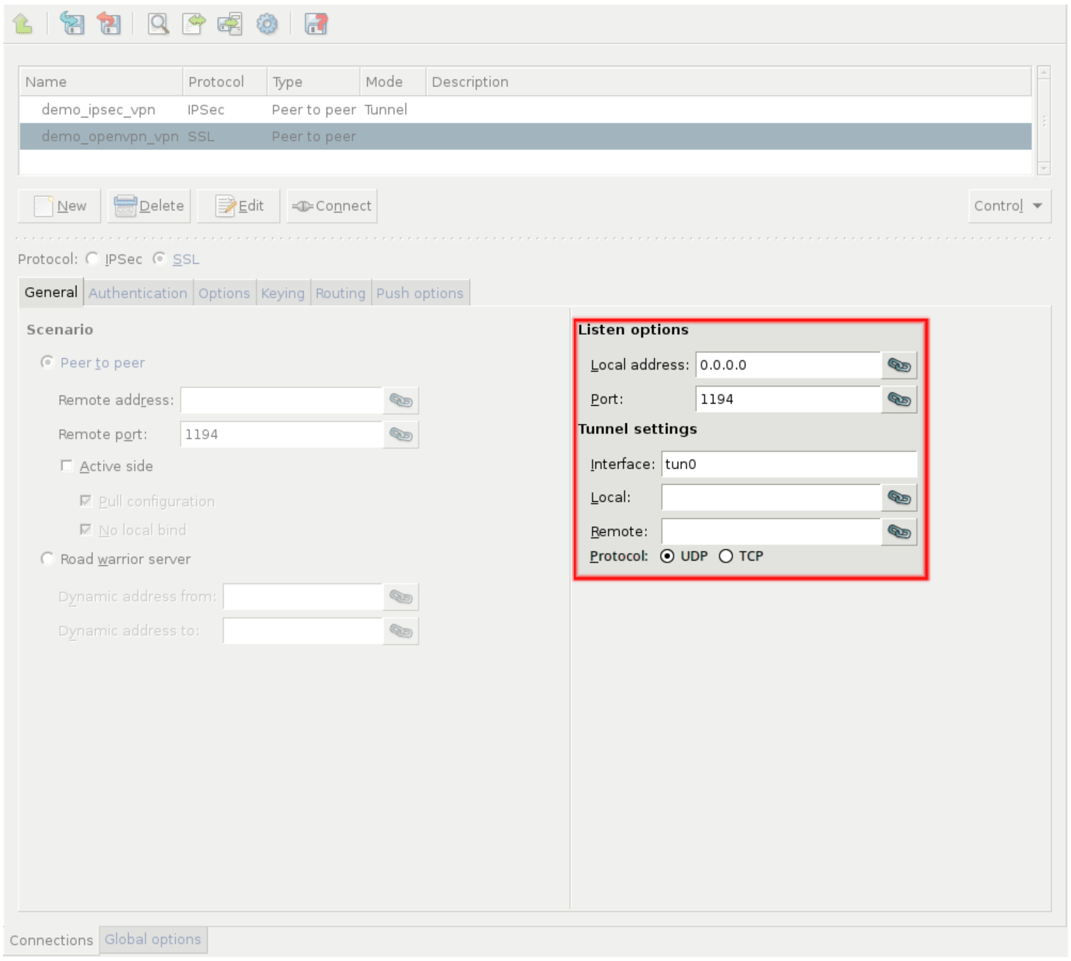
Figure 16.14. Selecting the SSL (OpenVPN) scenario

To create a Roadwarrior server, select the **Roadwarrior server** option.

Select the **Peer to Peer** option for other topologies.

Note
When creating a Network-to-Network connection, the two endpoints of the VPN tunnel are not used to communicate with each other. To encrypt the communication of the endpoints, create a separate Peer-to-Peer connection.

Step 5. Configure the local networking parameters. These parameters affect the Zorp endpoint of the VPN connection.



Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Protocol: [IPSec ✓ SSL]

Scenario: [✓ Peer to peer] Road warrior server

Remote port: 1194

Listen options

Local address: 0.0.0.0

Port: 1194

Tunnel settings

Interface: tun0

Protocol: [✓ UDP] TCP

Figure 16.15. Configuring local networking parameters

Set the following parameters in the **Listen options** section:

- **Local address**: Select the IP address that Zorp will use for the VPN connection. If Zorp should accept incoming VPN connections on every interface, enter the *0.0.0.0* IP address.
- **Port**: Zorp uses the port to listen for incoming VPN connections. Use the default port (*1194*) if nothing restricts that.

Note
These parameters have no effect if Zorp is the client-side of a VPN tunnel and does not accept incoming VPN connections.

Set the following parameters in the **Tunnel settings** section:

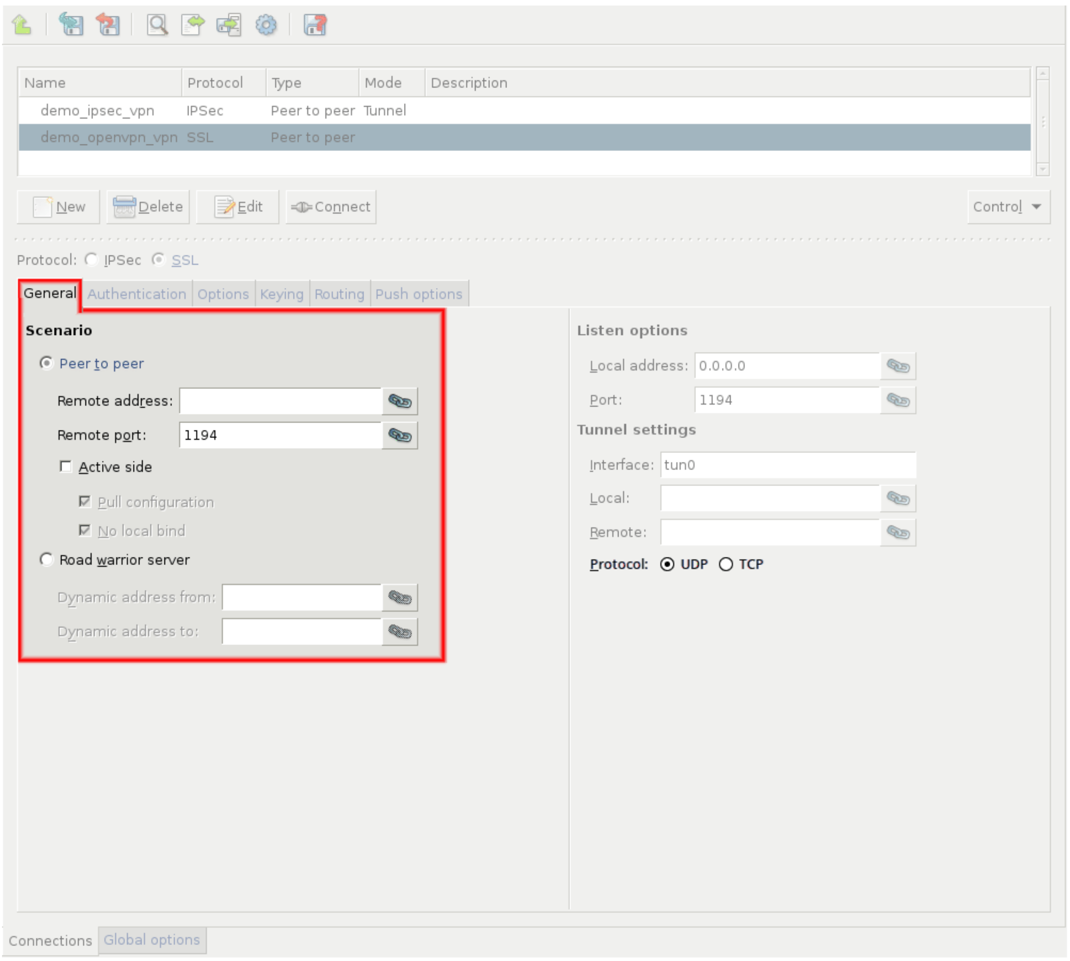
- **Interface**: It is the name of the virtual interface used for the VPN connection. ZMS automatically assigns the next available interface.
- **Local**: It denotes the IP address of Zorp as seen from the VPN tunnel. The *tun* interface will bind to this address, so Zorp rules can use this address.

- **Remote**: It is the IP address of the remote endpoint as seen from the VPN tunnel.
- By default, the VPN connections use the UDP protocol to communicate with the peers. To use the TCP protocol instead, select **Protocol** > **TCP**.

The **Local** and **Remote** addresses must be non-routable virtual IP addresses (for example, from the *`192.168.0 0`* range). These IP addresses are visible only on the *`tun`* interface, and are needed for building the VPN tunnel.

Warning
The **Local** and **Remote** addresses must be specified even for roadwarrior scenarios. Use the first two addresses of the dynamic IP range used for the remote clients.

Step 6. Configure the networking parameters of the remote endpoint.



Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Protocol: IPSec [✓ SSL]

Scenario

[✓ Peer to peer]

Remote address:

Remote port: 1194

☐ Active side

☑ Pull configuration

☑ No local bind

Road warrior server

Dynamic address from:

Dynamic address to:

Listen options

Local address: 0.0.0.0

Port: 1194

Tunnel settings

Interface: tun0

Local:

Remote:

Protocol: [✓ UDP] TCP

Figure 16.16. Configuring remote networking parameters

For Peer-to-Peer scenarios, set the following parameters:

- **Remote address**: It denotes the IP address of the remote endpoint.

- **Remote port**: Zorp connects this port to the remote VPN server. Use the default port (*1194*) if nothing restricts that.
- **Pull configuration**: Download the configuration from the remote endpoint. (Works only if the remote endpoint has its push options specified.)
- **No local bind**: Select this option if the Zorp host that is being configured shall run in client-mode only, without accepting incoming VPN connections.

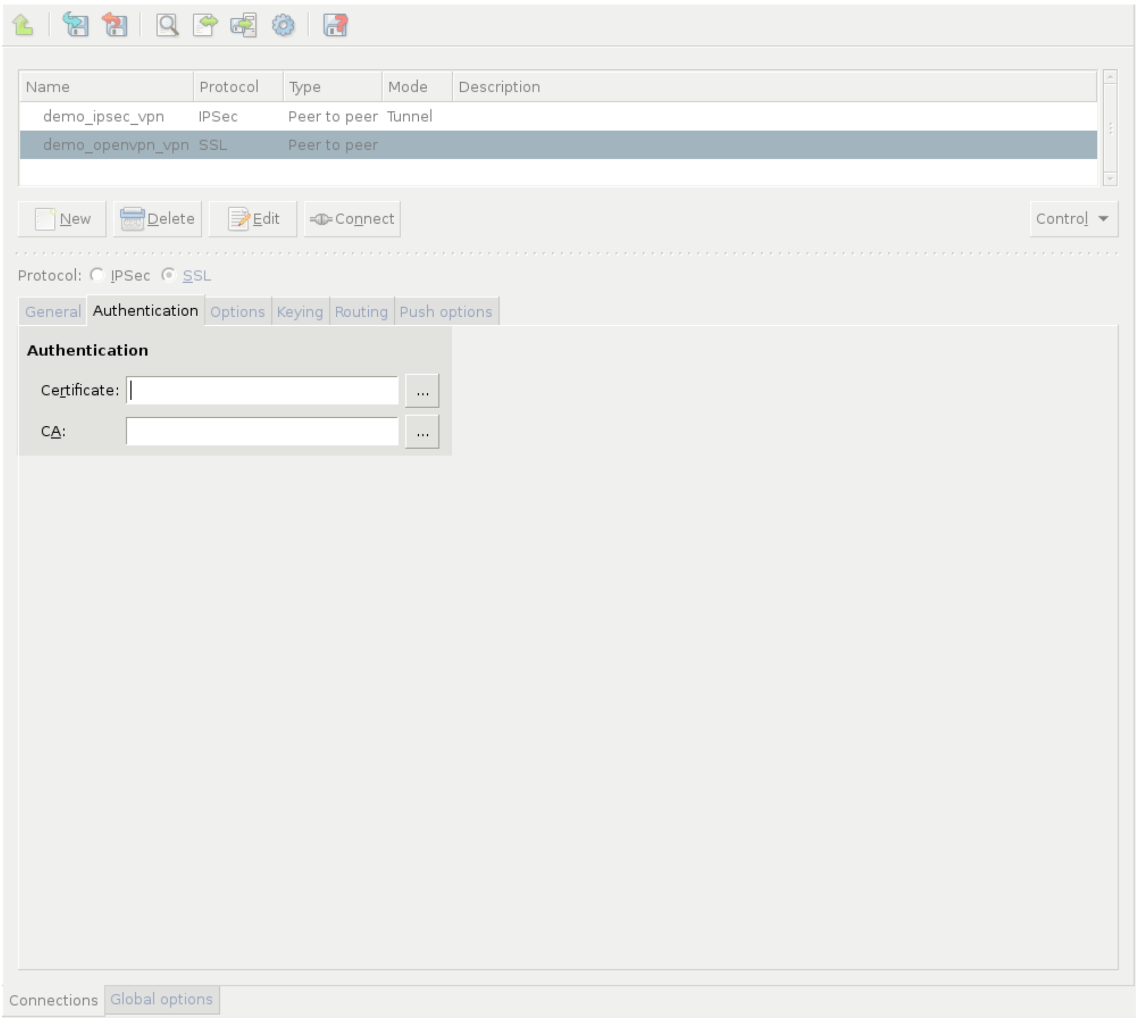
When Zorp acts as a roadwarrior server, set the IP address range using the **Dynamic address from** and **Dynamic address to** fields. Clients connecting to Zorp will receive their IP addresses from this range.

Note
The configured address range cannot contain more than 65535 IP addresses.

Every Windows client needs a */30* netmask (4 IP addresses). Make sure to increase the available address range when there are many Windows clients.

Step 7. When configuring Peer-to-Peer or Network-to-Network connections, select the **Active side** option so that Zorp initiates the VPN connection to the remote endpoint. If possible, enable this option on the remote endpoint as well.

Step 8. Click on the **Authentication** tab and configure authentication.



Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Authentication

Certificate:

CA:

Figure 16.17. Configuring authentication

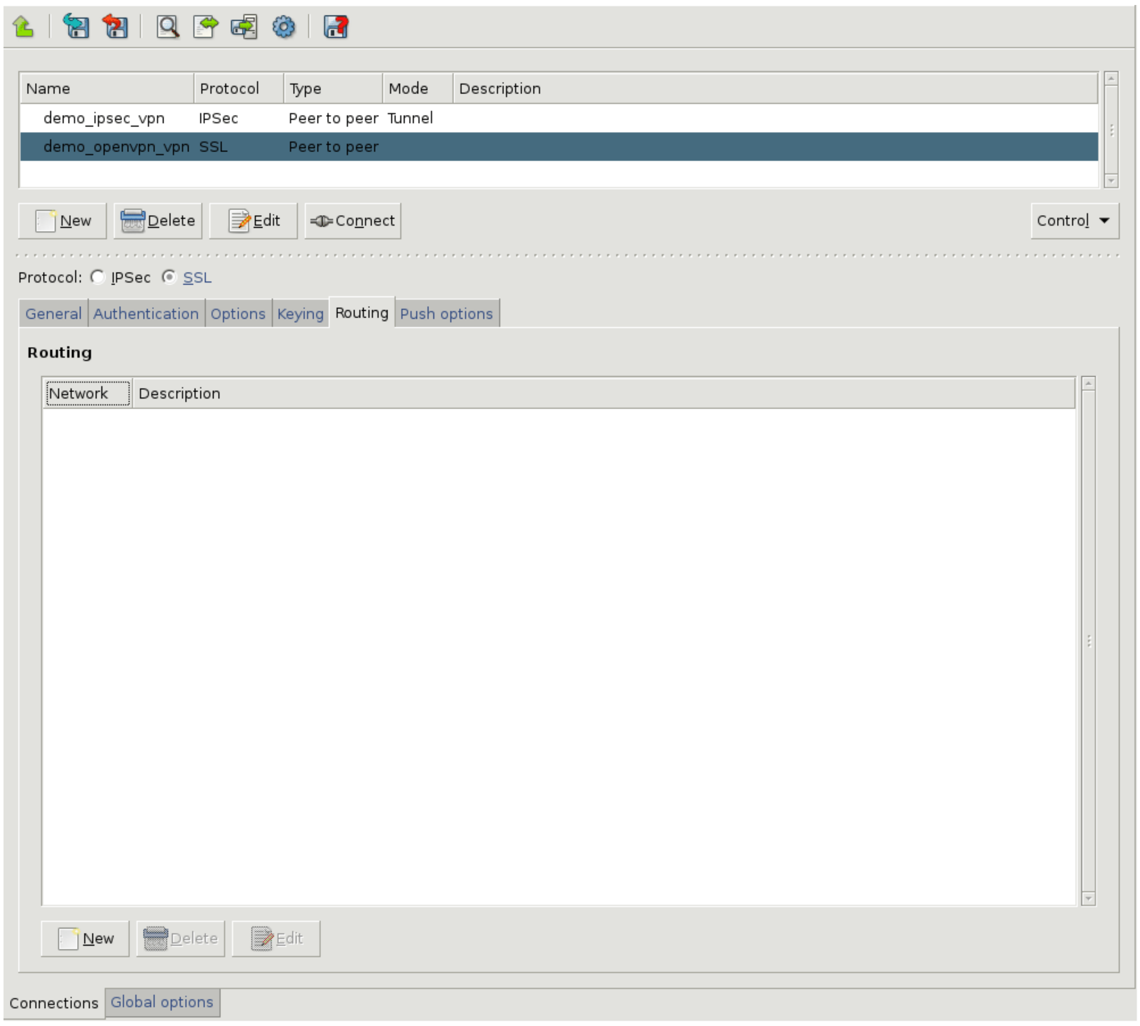
Set the following parameters:

- **Certificate**: Select a certificate available on the Zorp host. Zorp will show this certificate to the remote endpoint.
- **CA**: Select the trusted (Certificate Authority) CA group that includes the certificate of the root CA that issued the certificate of the remote endpoint. Zorp will use this CA group to verify the certificate of the remote endpoint.

Warning
If several remote endpoints use the same certificate to authenticate, only one of them can be connected to Zorp at the same time.

Note
See *Chapter 11, Key and certificate management in Zorp (p. 249)* for details on creating and importing certificates, CAs, and trusted CA groups required for certificate-based authentication.

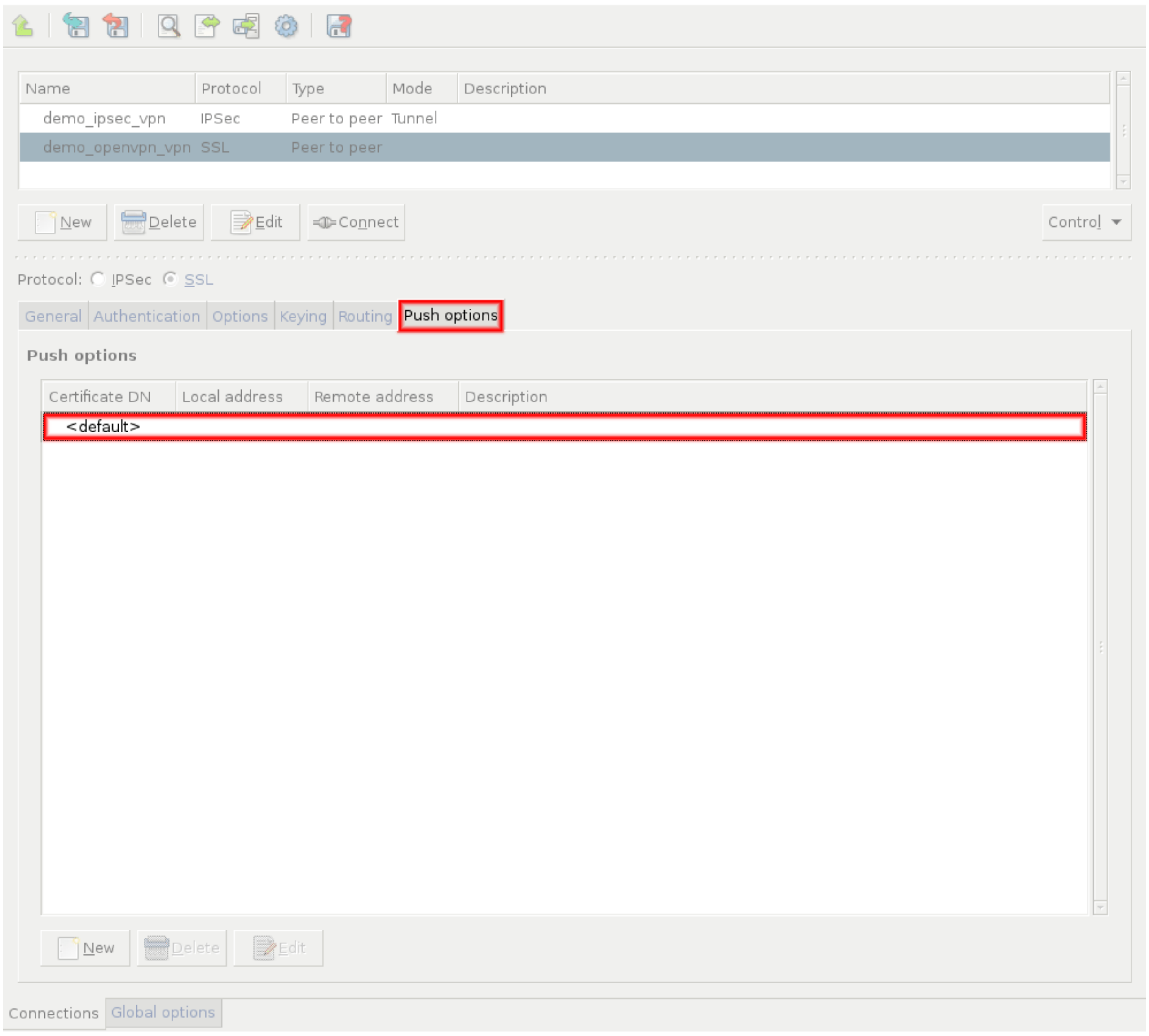
Step 9. Configure routing for the VPN tunnel. Click on the **Routing** tab, and add a routing entry for every network that is on the remote end of the VPN tunnel (or located behind the remote endpoint). Zorp sends every packet that target these networks through the VPN tunnel. To add a new network, click **New**, and enter the IP address and the netmask of the network.



Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Figure 16.18. Configuring tunnel routing

Step 10. Configure push options on the **Push options** tab.



Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Protocol: IPSec [✓ SSL]

Push options

Certificate DN	Local address	Remote address	Description
<default>			

Figure 16.19. Configuring push options

Tip
Push options are most often used to set the configuration of roadwarrior clients. For example, it can be used to assign a fix IP address to a specific client.

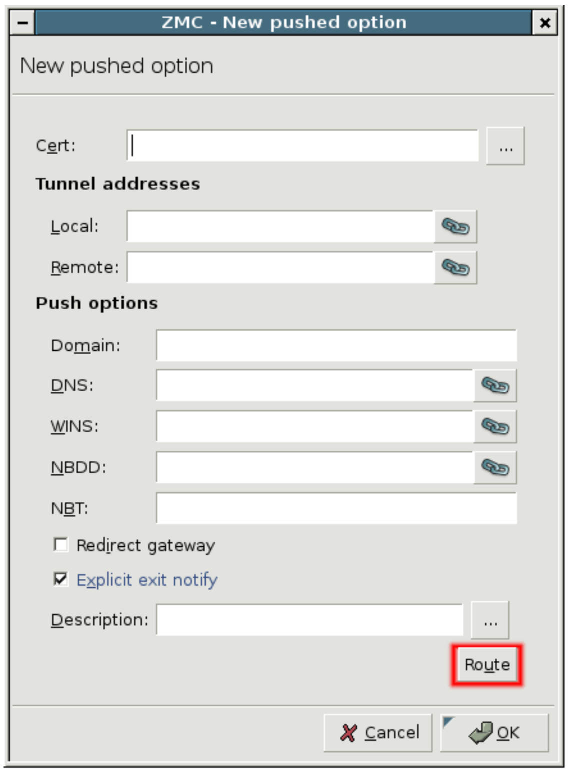


Figure 16.20. Configuring client routing

Click **Route** to add routing entries for the remote endpoint. These routing entries determine which networks protected by Zorp are accessible from the remote endpoint.

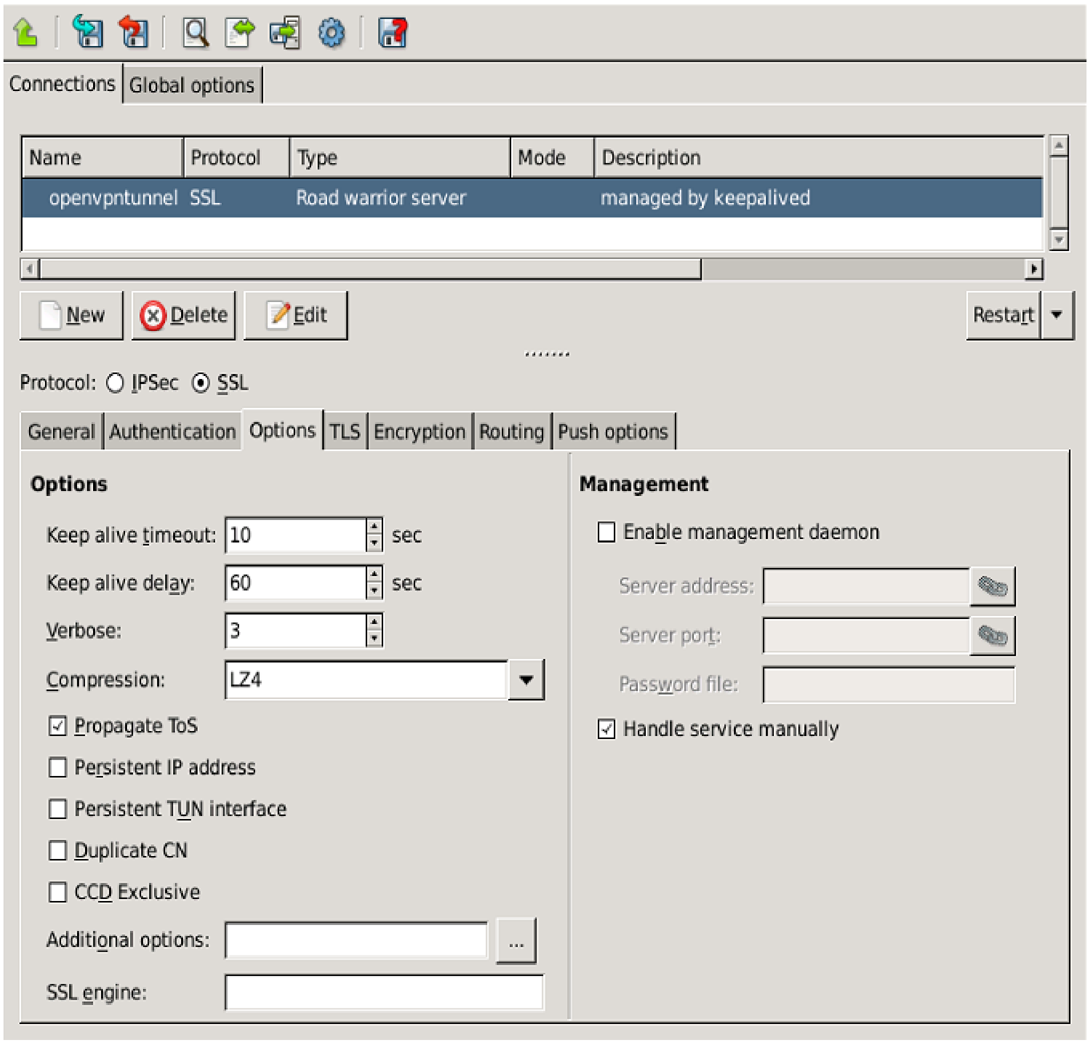
See *Section 16.4.3.2, Push options (p. 453)* for details.

Step 11. Set other options as needed. See *Section 16.4.3, SSL options (p. 450)* for details.

16.4.3. SSL options

Special options of a particular SSL VPN connection can be set on the **Options** and the **Keying** tabs.

Note
Do not modify these options unless it is a must and the required expertise is available.


Connections | Global options

Name	Protocol	Type	Mode	Description
openvpntunnel	SSL	Road warrior server		managed by keepalived

Protocol: IPSec [✓ SSL]

General | Authentication | Options | TLS | Encryption | Routing | Push options

Options

Keep alive timeout: 10 sec

Keep alive delay: 60 sec

Verbose: 3

Compression: LZ4

☑ Propagate ToS

☐ Persistent IP address

☐ Persistent TUN interface

☐ Duplicate CN

☐ CCD Exclusive

Additional options:

SSL engine:

Management

☐ Enable management daemon

Server address:

Server port:

Password file:

☑ Handle service manually

Figure 16.21. Configuring OpenVPN options

The following options can be set on the **Options** tab:

- **Keep-alive timeout**: Zorp pings the remote endpoint periodically. This parameter specifies the time between two ping messages in seconds.
- **Keep-alive delay**: The amount of time in seconds until Zorp waits for a response to the ping messages. If no response is received within this period, Zorp restarts the VPN connection.
- **Verbose**: It is the verbosity level of the VPN tunnel.
- **Compression**: Compress the data transferred in the VPN tunnel.
- **Propagate ToS**: If enabled and the Type of Service (ToS) parameter of the packet transferred using the VPN is set, Zorp sets the ToS parameter of the encrypted packet to the same value.
- **Persistent IP address**: **This option is available only in Zorp 3.3R6 or later.** Preserve the initially resolved local IP address and the port number across SIGUSR1 or --ping-restart restarts.
- **Persistent TUN Interface**: **This option is available only in Zorp 3.3R6 or later.** Create a persistent tunnel. Normally TUN/TAP tunnels exist only for the period of time that an application has them

open. Enabling this option builds persistent tunnels that live through multiple instantiations of OpenVPN and die only when they are deleted or the machine is rebooted.

- **Duplicate CN**: **This option is available only in Zorp 3.4 or later.** If enabled, multiple clients with the same common name can connect at the same time. If this option is disabled, Zorp will disconnect new clients if a client having the same common name is already connected.
- **CCD Exclusive**: **This option is available only in Zorp 3.4 or later.** If enabled, the connecting clients must have a `--client-config-dir` file configured, otherwise the authentication of the client will fail. This file is generated automatically if the **Roadwarrior Server** option is enabled on the **General** tab.
- **Additional options**: **This option is available only in Zorp 3.4 or later.** Enter any additional options required to be set here. Options entered here are automatically appended to the end of the configuration file of the VPN tunnel.
- **SSL engine**: Use the specified SSL-accelerator engine.
- **Enable management daemon**: Enable a TCP server on an IP port to handle daemon management funtions. The password provided is used by the TCP clients to access management functions.
  While the management port is designed for the programmatic control of the OpenVPN by other applications, it is possible to telnet to the port, using a telnet client in *raw* mode. Once connected, type *help* for a list of commands.
- **Handle service manually**: Do not start this VPN at boot (omit from the /etc/default/openvpn file). This VPN will be managed by other processes like by keepalived or by monitoring. This tunnel will not be accidentally started or stopped with the global control button.

The options of the **Keying** tab specify the encryption used in the connection. Modify these parameters only if it is necessary for compatibility with the remote endpoint.

16.4.3.1. Procedure – Configuring the VPN management daemon

This option is available only in Zorp 3.3R6 or later. For details on the OpenVPN management interface, see *the management-notes.txt file in the management folder of the OpenVPN source distribution*.

Step 1. To enable the management daemon for a particular SSL VPN connection, select the **VPN** ZMC component, select the particular SSL connection, and click the **Options** tab.

Step 2. Select **Enable management daemon**.

Step 3. Enter the IP address where the daemon will accept management connections into the **Server address** field. It is strongly recommended that IP is set to `127.0.0.1` (localhost) to restrict accessibility of the management server to local clients.

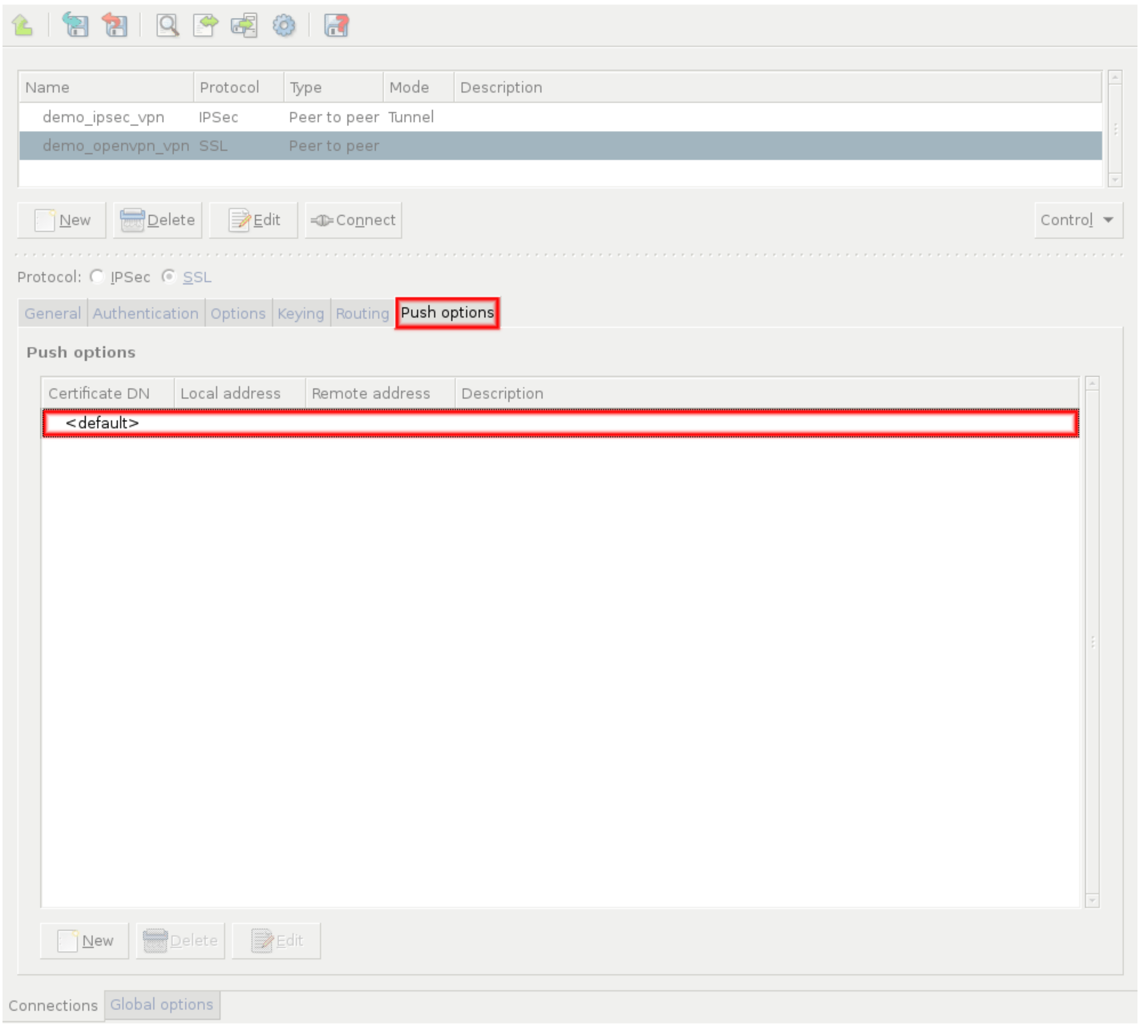
Step 4. Enter the port number where the daemon will accept management connections into the **Server port** field. Note that the IP address:port pair must be unique for each management interface.

Step 5. Set the path to a file that will store the password to the management daemon. Clients connecting to the management interface will be required to enter the password set in the first line of the password file.

Step 6. Save the changes and repeat the above steps for other VPN connections if needed.

16.4.3.2. Push options

Push options are settings that the remote clients can download from Zorp when the VPN tunnel is built.



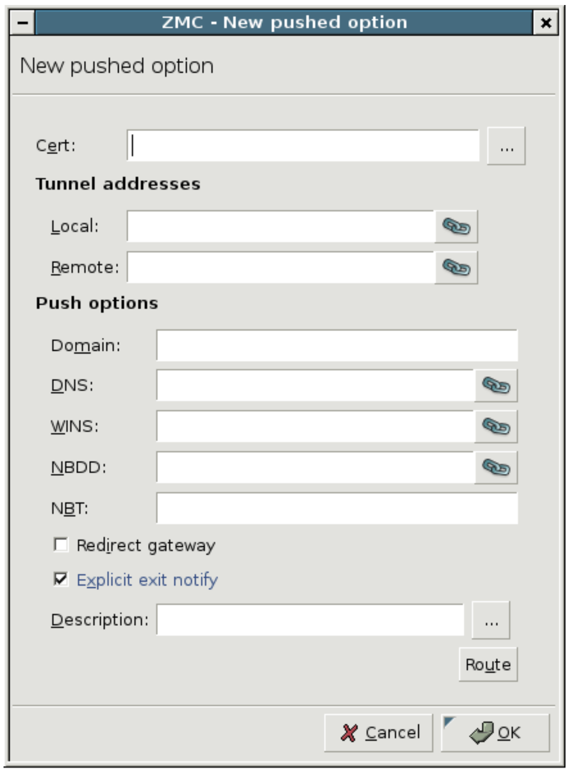
Name	Protocol	Type	Mode	Description
demo_ipsec_vpn	IPSec	Peer to peer	Tunnel	
demo_openvpn_vpn	SSL	Peer to peer		

Push options

Certificate DN	Local address	Remote address	Description
<default>			

Figure 16.22. Configuring global push options

To set push options that apply for every remote endpoint of the selected VPN connection, double-click the **<default>** entry.


ZMC - New pushed option

New pushed option

Cert:

Tunnel addresses

Local:

Remote:

Push options

Domain:

DNS:

WINS:

NBDD:

NBT:

☐ Redirect gateway

☑ Explicit exit notify

Description:

Route

Cancel OK

Figure 16.23. Configuring push options

The following push options can be set on the **Push options** tab:

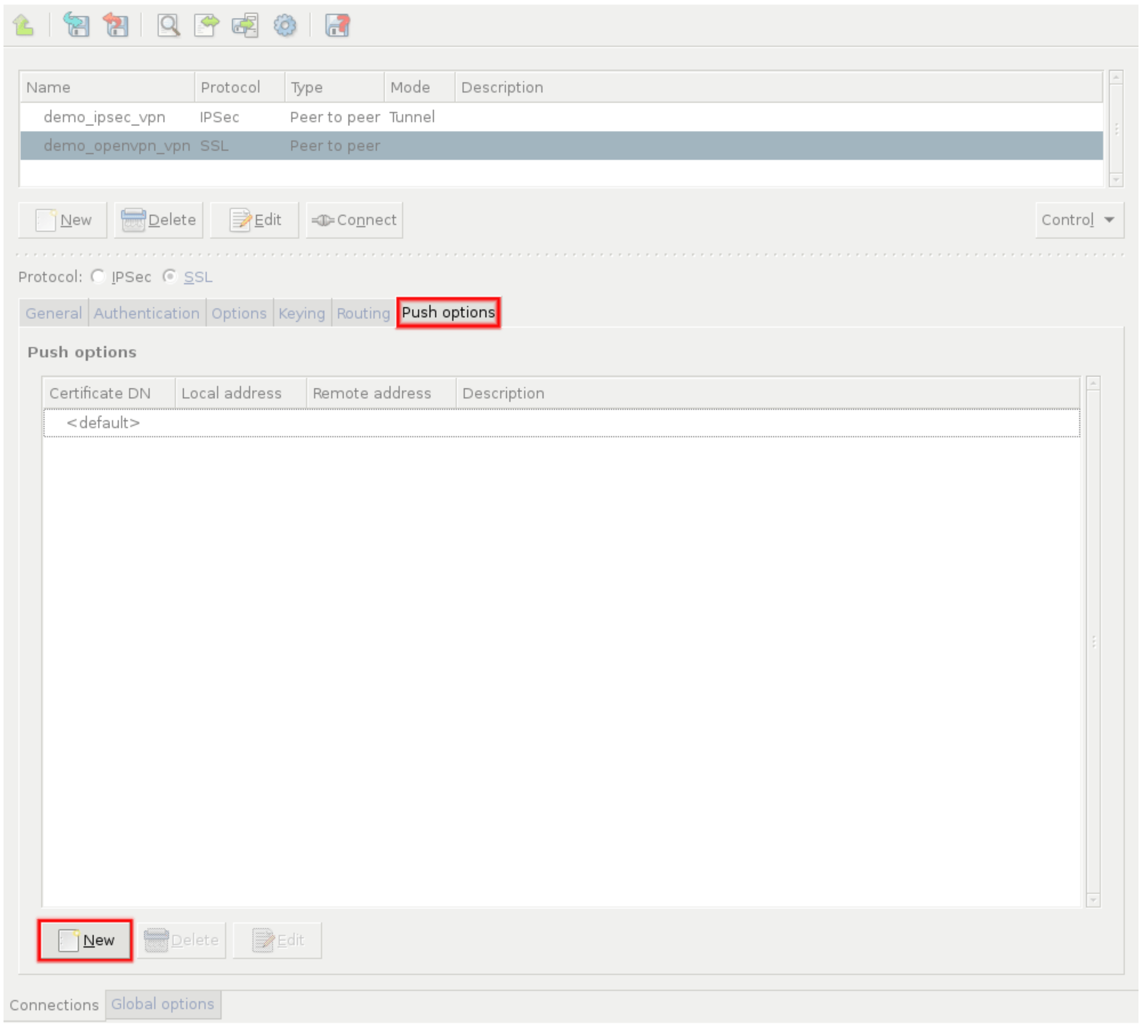
- **Domain**: It is the domain of the network.
- **DNS**: It denotes the address of the Domain Name Server (DNS).
- **WINS**: It is the address of the Windows Internet Name Service (WINS) Server.
- **NBDD**: It is the address of the NetBIOS Datagram Distribution (NBDD) Server.
- **NBT**: It is the type of the NetBIOS over TCP/IP node. Enter the number corresponding to the selected mode:
  - *1*: Send broadcast messages.
  - *2*: Send point-to-point name queries to a WINS server.
  - *4*: Send broadcast message and then query the nameserver.
  - *8*: Query name server and then send broadcast message.
- **Redirect gateway**: It sends every network traffic of the remote endpoint through the VPN tunnel. See *Section The Redirect gateway option (p. 457)* for details.

Note
Using the **Redirect gateway** option means that the remote client will have access only to the services permitted by Zorp for the VPN tunnel when the VPN tunnel is active. For example, the client will not be able to surf the Internet using HTTP if Zorp allows only POP3 services for the clients connected using the VPN.

- **Explicit exit notify**: The remote endpoint sends a message to Zorp before closing the VPN tunnel. If this option is disabled, Zorp does not immediately notice that an endpoint became unavailable, and error messages might appear in the Zorp logs.
- **Additional options**: **This option is available only in Zorp 3.4 or later.** Enter any additional push options that need to be set here. Options entered here are automatically appended to the end of the `.ccd` file of the VPN tunnel. This option can be used for example to set the *`iroute`* parameter.
- **Route**: Add routing entries for the remote endpoint. These routing entries determine which networks protected by Zorp are accessible from the remote endpoint.

To set push options for a specific remote endpoint, click **New** and select the certificate of the remote endpoint.

Note
Alternatively, enter the Unique Name of the endpoint certificate into the **Cert** field. That way, certificates not available in the Zorp PKI system can be used as well. **This option is available only in Zorp 3.4 or later.**



Name | Protocol | Type | Mode | Description
demo_ipsec_vpn | IPSec | Peer to peer | Tunnel
demo_openvpn_vpn | SSL | Peer to peer
New
Delete
Edit
Connect
Control
Protocol: IPSec SSL
General
Authentication
Options
Keying
Routing
Push options
Push options
Certificate DN | Local address | Remote address | Description
<default>
New
Delete
Edit
Connections
Global options

Figure 16.24. Configuring client-specific push options

In this case, the IP addresses visible in the tunnel can also be set, so a fixed IP address can be assigned to the client using the **Local** parameter. Note that the **Local** and **Remote** directions are from the client's perspective: **Local** is the remote client's IP address in the VPN tunnel, while **Remote** is the IP address of Zorp in the VPN tunnel.

When assigning fixed IP addresses to Windows clients, remember that every Windows client needs a `/30` netmask (4 IP addresses). For every client, use an IP pair of the following list as the last octet of the **Local** and **Remote** IP addresses:

```
[  1,  2] [  5,  6] [  9, 10] [ 13, 14] [ 17, 18]
[ 21, 22] [ 25, 26] [ 29, 30] [ 33, 34] [ 37, 38]
[ 41, 42] [ 45, 46] [ 49, 50] [ 53, 54] [ 57, 58]
[ 61, 62] [ 65, 66] [ 69, 70] [ 73, 74] [ 77, 78]
[ 81, 82] [ 85, 86] [ 89, 90] [ 93, 94] [ 97, 98]
[101,102] [105,106] [109,110] [113,114] [117,118]
[121,122] [125,126] [129,130] [133,134] [137,138]
[141,142] [145,146] [149,150] [153,154] [157,158]
[161,162] [165,166] [169,170] [173,174] [177,178]
```

```
[181,182] [185,186] [189,190] [193,194] [197,198]
[201,202] [205,206] [209,210] [213,214] [217,218]
[221,222] [225,226] [229,230] [233,234] [237,238]
[241,242] [245,246] [249,250] [253,254]
```

The Redirect gateway option

Enabling the **Redirect gateway** push-option overrides the default gateway settings of the remote endpoint and sends every network traffic of the remote endpoint through the VPN tunnel. The remote endpoint can only access the Internet through the VPN tunnel. That way Zorp can control what kind of communication (protocols, and so on) can the remote client use while connected to the internal network using the VPN tunnel.

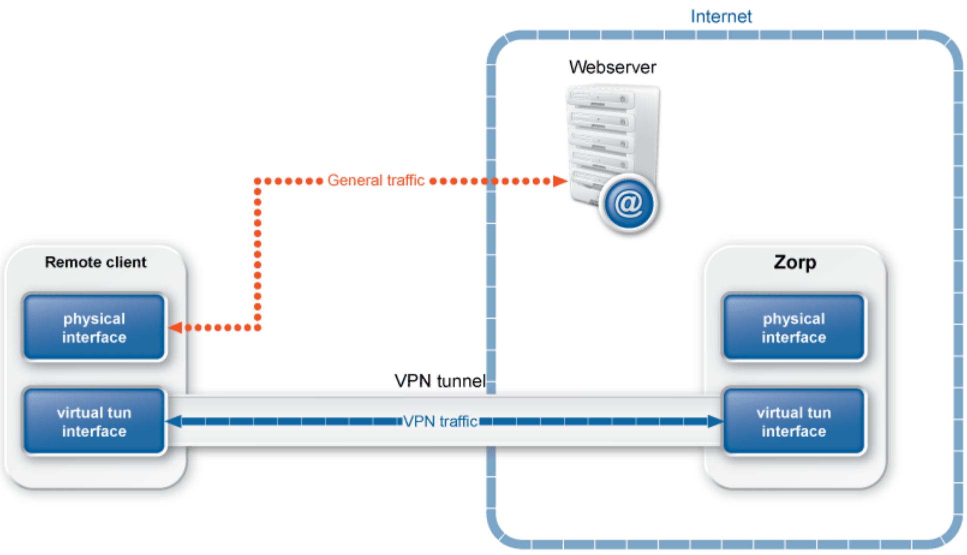


Figure 16.25. Normal routing

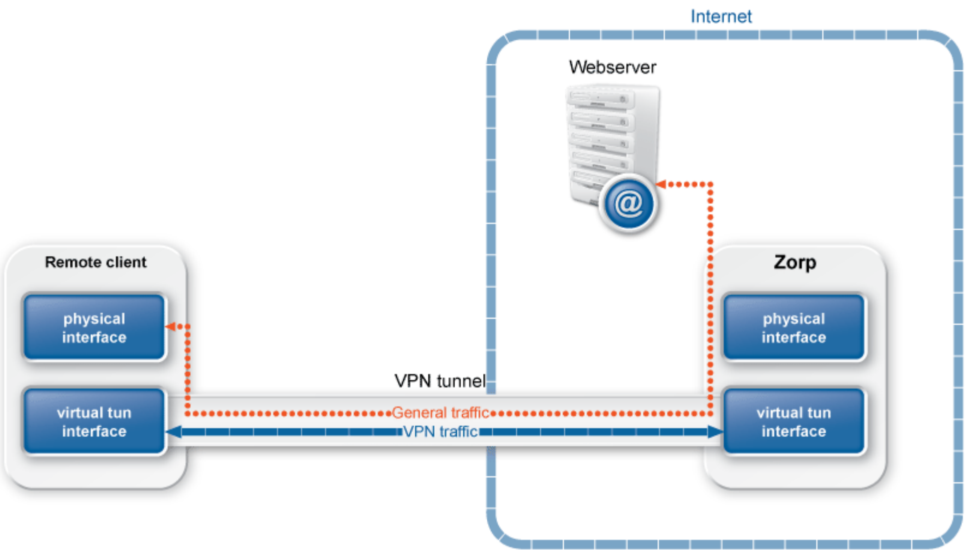


Figure 16.26. Using the Redirect gateway option

The following flags can be set for the **Redirect gateway** option, with the **Def1** being set as default:**Redirect gateway flags are available only in Zorp 3.4 or later.**

- **Local**: Select this option if the end-points of the VPN tunnel are directly connected through a common subnet, such as wireless. Note that in this case Zorp does not create a static route for the remote address of the tunnel.
- **Bypass DHCP**: Select this option to add a direct route to the DHCP server (if it is non-local) which bypasses the VPN tunnel.
- **Def1**: Select this option to override the default gateway by using `0.0.0.0/1` and `128.0.0.0/1` instead of `0.0.0.0/0`. That way the original default gateway is overridden but not deleted.
- **Bypass DNS**: Select this option to add a direct route to the DNS server(s) (if it is non-local) which bypasses the VPN tunnel.

Chapter 17. Integrating Zorp to external monitoring systems

Zorp Professional 7 allows the administrator to monitor the resources of the Zorp hosts using external monitoring tools. Zorp can be integrated to the following monitoring environments:

- *Munin*
- *Nagios*

17.1. Procedure – Monitoring Zorp with Munin

Purpose:

To monitor the Zorp Professional hosts with Munin, complete the following steps. Using Munin, it is possible to monitor the memory usage of the hosts, and the number of running processes and threads for the Zorp instances.

Steps:

Step 1. Login to the host locally, or remotely using SSH. For details on enabling SSH access, see *Section 9.4, Local services on Zorp (p. 232)*.

Step 2. Install the required packages using the `sudo apt install <package-name>` command. Which package has to be installed depends on the role of the host.

- For Zorp hosts, install the *`zorp-pro-munin-plugins`* package.
- For ZMS hosts, install the *`zms-munin-plugins`* package.
- For ZCV hosts, install the *`zcv-munin-plugins`* package.
- If a host has multiple roles, install every applicable package. For example, if ZMS and ZCV are running on the same host, install the *`zms-munin-plugins`* and *`zcv-munin-plugins`* packages.

Step 3. Install the *`munin-node`* package and configure it according to the needs of the environment. For details, see the *Munin documentation*.

Step 4. Login to your ZMS host using ZMC, and enable access to the *`TCP/4949`* port. For details, see *Section 9.4, Local services on Zorp (p. 232)*.

Step 5. Repeat this procedure for each Zorp Professional host that is required to be integrated into the monitoring system.

17.2. Procedure – Installing a Munin server on a ZMS host

Purpose:

If there is no separate Munin server, and Zorp is not preferred to be integrated into the existing Munin environment, install the Munin server on a standalone ZMS host (installing the Munin server on a Zorp or ZCV host is possible, but strongly discouraged). To achieve this, complete the following steps.

Steps:

Step 1. Login to the host locally, or remotely using SSH. For details on enabling SSH access, see *Section 9.4, Local services on Zorp (p. 232)*.

Step 2. Install the Munin server and the *Lighthttpd* webserver packages. The webserver is required to display the resource graphs collected using Munin. Issue the following command: `sudo apt install munin lighttpd`.

Step 3. Configure the Munin server and the webserver as needed for the actual environment. For details, see the documentation of *Munin* and *Lighthttpd*.

Warning

- By default, access to the Munin graphs does not require authentication.
- Configure Munin and *Lighthttpd* to use SSL-encryption, and disable unencrypted HTTP access on port 80. Use port 443, or a non-standard port instead.

Step 4. Login to the ZMS host using ZMC, and enable access to the TCP port configured in the previous step on the ZMS host. For details, see *Section 9.4, Local services on Zorp (p. 232)*.

17.3. Procedure – Monitoring Zorp with Nagios

Purpose:

To monitor Zorp Professional hosts with Nagios, complete the following steps.

Using Nagios, the following details can be monitored:

- the memory usage of the hosts
- the number of running processes and threads for the Zorp instances
- the expiry of the product licenses and certificates (for details on certificate and license-monitoring, see *Procedure 11.3.8.8, Monitoring licenses and certificates (p. 286)*)

Prerequisites:

- To monitor the Zorp Professional hosts with Nagios, there must already be a central Nagios server installed. It is not possible to install the Nagios server on Zorp Professional hosts.
- Experience in administering Nagios is required.

Steps:

Step 1. Login to the host locally, or remotely using SSH. For details on enabling SSH access, see *Section 9.4, Local services on Zorp (p. 232)*.

Step 2. Issue the following command to install the required packages: `sudo apt install zorp-pro-nagios-plugins nagios-nrpe-server`. The *`zorp-pro-nagios-plugins`* package installs three scripts; these are automatically configured to run as root, and are listed in the `/etc/nagios/nrpe.d/zorp.cfg` file.

Step 3. Login to your ZMS host using ZMC, and enable access to the *`TCP/5666`* port. For details, see *Section 9.4, Local services on Zorp (p. 232)*.

Step 4. Repeat this procedure for every Zorp Professional host that is required to be integrated into the actual monitoring system.

Step 5. Add the Zorp Professional hosts to the central Nagios server, and create services for the hosts. For details, see the *documentation of Nagios*.

> **Note**
> Adjust the alerting limits set in the scripts as needed for the environment.

Appendix A. Packet Filtering

As of Zorp 3.3, packet filtering and application-level services are handled together, consequently these topics are discussed together in *Chapter 6, Managing network traffic with Zorp (p. 87)*. Manually modifying the packet filtering rules is required only very rarely, and is not recommended unless absolutely needed. Local Zorp services are described in *Section 9.4, Local services on Zorp (p. 232)*.

The key point of the Zorp firewall system is the Zorp-based application proxy suit. Besides the application layer gateways, the enclosed packet filter also plays a very important role. Although all of the traffic is handled by the Zorp proxies, the packet filter also performs additional filtering and helps the proxies' work.

This chapter includes a short introduction on packet filter basics and technologies in general and also shows the main concepts of the Linux packet filter framework which is used with Zorp. It also covers the commonly used packet filter policy style which is the default of the ZMS-based configuration. For further reading on the Linux packet filter, see *Appendix C, Further readings (p. 485)*.

In the world of computer networks each and every connection is based on packets. No communication takes place without packets. Therefore if the filter of the traffic (connections) is required, it is reasonable to filter the packets. Unlike proxies, packet filters operate with packets on the packet level. If the firewall drops the packets it would result in the drop of the connection.

Note
Packet filtering rules are created and managed automatically by ZMS. Usually it is not required nor recommended to modify them manually. If the transfer of traffic is required without application-level inspection, create a packet filter service (see *Procedure 6.4.1, Creating a new service (p. 115)* for details). To enable access to services running on firewall hosts (e.g., SSH access), see *Section 9.4, Local services on Zorp (p. 232)*.

Typically, the packet filtering rules have to be modified when traffic without terminating it on Zorp has to be forwarded, like forwarding IPSec VPN connections.

A.1. How packet filtering works

As packet filtering works with individual packets, a decision is needed for each and every packet whether that specific packet can pass or should undergo some other action. Packet filtering can work with incoming and outgoing packets as well, but the basic decision making procedure is the same. The basic steps of packet filtering are the following.

1. The filter system inspects the packet. It usually checks the following information in the packet header:
   - source/destination IP addresses
   - IP options
   - TOS/TTL fields
   - source/destination port numbers
   - TCP flags
   - data part of packet

- and so on

  Stateful packet filtering can check the state of the given packet related to the known connections (whether this packet belongs to an already seen connection or it is a new packet or it is a packet related to an already established connection, like an ICMP control packet), or to other stateful information (whether this packet fits in the TCP window of the connection). This piece of information is necessary for the decision. Also, the firewall can check whether the packets' checksum and the packets in general are adequate.

2. After inspecting the packet and collecting stateful information, the packet filter evaluates the policy for the given packet. The policy and the representation of the policy might be different for various implementations, but usually Access Control Lists (ACL) are used. ACLs are checked from the top of the list to the bottom. The list entries are usually called rules and these rules are evaluated after one another.
   A rule usually contains a match and a verdict part. The match part is evaluated based on the information gathered from the packet before the policy check. If a packet matches the rule the rule's verdict is taken for that packet. The various implementations differ in how they run through the list. One stops evaluating at the first match, while others might take the last match's verdict.

3. After evaluating the ACL the packet filter can work with that packet according to the verdict. Usually, every ACL has a default verdict which controls what should happen with the packet if no match occurs. Based on the main security rule a default deny or default drop approach is a good choice, but as usual it depends on the implementation and on the administrator.
   There are numerous verdicts, but usually all implementations support the following basic verdicts. The meaning of the verdicts, though, might differ slightly.

   - Accept,
     allowing the packet to pass.

   - Deny/Drop,
     denying the packet silently meaning that no error packet is sent back to the sender.

   - Reject,
     denying the packet with sending back some kind of error packet (ICMP error message or TCP reset packet depending on the situation).

To successfully deploy and easily troubleshoot any packet filter firewall it is necessary to understand how the specific implementation handles the packets and how it evaluates its policy. It is also necessary to learn how the policy is represented and how the configuration is constructed. Being familiar with the deeper details of implementation helps in configuration.

A.2. Packet filtering on Linux

Zorp is based on Linux and like most modern operating systems it also has a packet filtering solution. The Linux kernel has had serious filtering capabilities since version 2.0. Since then, the packet filtering framework has been rewritten three times to improve its capabilities, features, speed and robustness. The latest packet filtering system in Linux is called Netfilter/IPTables and is available since version 2.4.

Netfilter belongs to the family of stateful packet filtering methods and provides packet mangling and connection NATing capabilities as well. Netfilter is designed to be very flexible in configuration to cover all of the possible packet filtering situations. Although in Zorp, Netfilter plays less significant role, it is necessary to understand how it handles packets and how the configuration is organized.

A.3. Understanding Netfilter and IPTables

Netfilter itself is a framework in the network subsystem of the Linux kernel. It allows packet filtering, network address translation, and various different packet mangling. It is very sophisticated and open enough to be able to satisfy all the potential needs, but it is a framework only. To be able to utilize its capabilities, an easily configurable policy layer is required. This can be realized by IPTables which can be found in the kernel as well. IPTables is built on the Netfilter framework and extensively uses it, that is why it is impossible to use IPTables without knowing the basics of the underlying framework.

In real life scenarios, IPTables and Netfilter work very closely together. It is nearly impossible to separate them. From administration point of view, they do not need to be separated. In Zorp, Netfilter works through the mediation of IPTables. Netfilter/IPTables is built from smaller blocks which cooperate with each other.

The key building blocks of Netfilter and their roles are presented in the following chapters.

A.3.1. Hooks

All incoming, outgoing and passing packets must go through the TCP/IP stack of the Linux kernel and as Netfilter works with packets, the best way to influence the packet flow is to cooperate with the protocol stack. At some point of the packet flow, the Netfilter framework hooks into the Linux kernel. These points are called hooks where the stack passes the packet to the framework, which evaluates the policy on the given packet.

Based on the result of the policy evaluation, the framework can response in the following three ways:

- gives back the packet to the stack to allow it to be processed further,
- gives back a modified packet,
- or drops the packet preventing it from further processing.

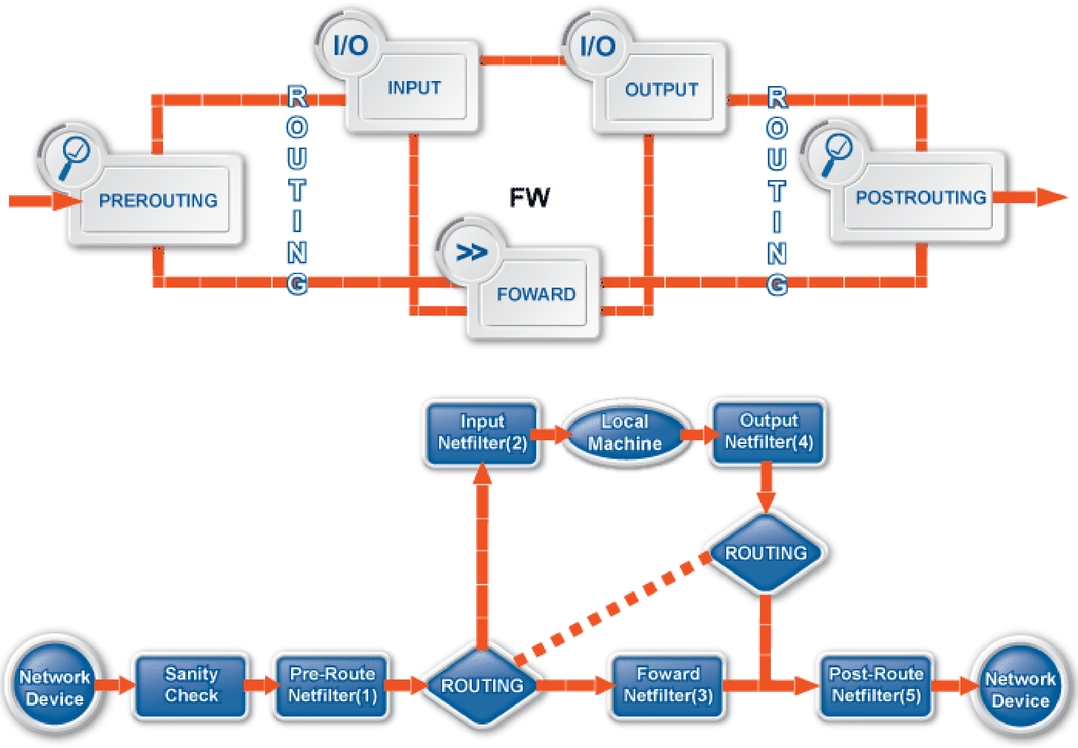


Figure A.1. Netfilter hooks

Currently, Netfilter defines five hooks in the kernel.

PREROUTING	Right after the arrival of the packet from the network and a simple sanity check but before any routing decision is made the packet is sent to this hook.
INPUT	If the destination of the packet (based on the routing decision) is the host itself, the packet is sent to this hook after the routing decision, but before the delivery of the packet to any application.
FORWARD	If the destination of the packet (based on the routing decision) is not the host itself and if forwarding is enabled, the packet is sent to this hook after the routing decision, but before leaving the host.
OUTPUT	If the packet is originating from the host and before any routing decision is made, the packet is sent to this hook.
POSTROUTING	If the packet is leaving the host and the routing decision is made, the packet is sent to this hook, but before actually passing the packet to the network interface for transmission.

Note
The packet has to pass all of the hooks to get to its destination. It is not enough to pass one hook, all of the hooks must be passed. Obviously, if the packet is dropped on one hook it cannot get to any other hooks. Its processing is ended where it was dropped.

For example, if the destination of the packet was the host, it has to successfully pass (be accepted) at the PREROUTING and the INPUT hooks. If the destination of the packet was not the host (so it is passing the host) it has to pass at the PREROUTING, FORWARD and the POSTROUTING hooks. The third case is when the packet leaves the box (originating from some application running on the host) it has to pass at the OUTPUT and the POSTROUTING hooks.

These hooks provide the Netfilter framework itself, but they are actually used by IPTables providing the heart of the Linux packet filter system this way.

A.3.2. Tables

Hooks only provide an interface where the packets are accessible, but they do not provide a solution to handle the packets. To manage the packets, the Linux kernel provides basically three options which are represented by tables.

- Filter table,
  responsible for Packet Filtering.

- NAT table,
  responsible for Network Address Translation.

- Mangle table,
  responsible for Packet Mangling.

Each table is responsible for one specific action. To fulfil their roles, the tables need to access the packets. Therefore, the tables are registered in the hooks so that they can influence the packet flow in the kernel.

The tables can be configured to carry out other tables' tasks though this solution is not recommended since the tables perform different things and they have different requirements from the hooks, accordingly. Some of the tables cannot be tied to all hooks, but it is also possible that more than one table is involved in one hook. Consequently, more tables can share packets on one hook. Each table can be registered to any hook with a specific priority. The order in which the tables receive the passing packet is based on this priority list.

To successfully pass a hook, the packets have to be accepted on each and every table registered to that given hook. If a packet is dropped by any of the tables it cannot get to the subsequent tables, nor to the subsequent hook.

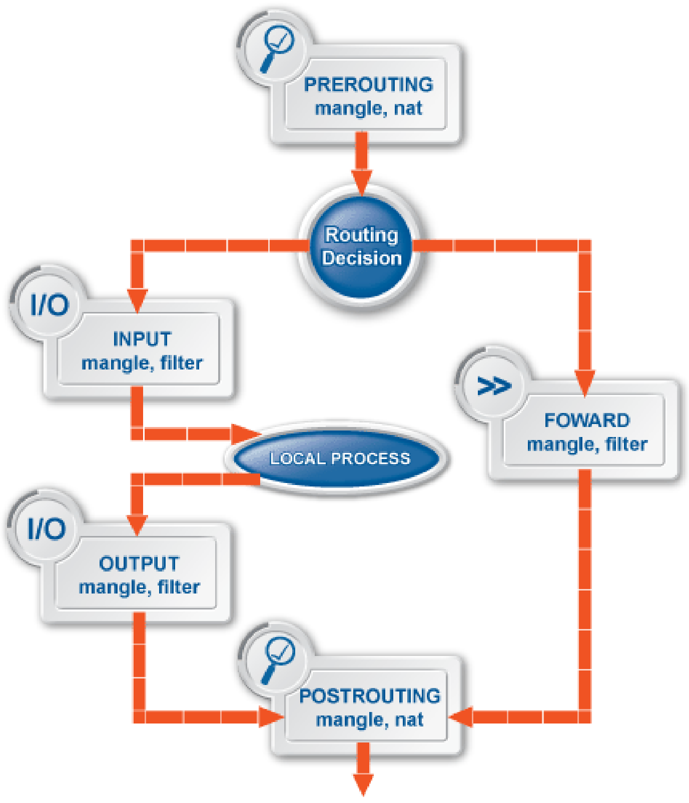


Figure A.2. Netfilter tables

The following tables are involved in packet filtering:

filter table	This table is responsible for the packet filtering, so it represents the classic packet filtering items. Packets are usually dropped, rejected and accepted here. This table is registered to the INPUT, FORWARD and to the OUTPUT hooks with the highest priority, meaning that it is the last table which gets the packets at these hooks.
NAT table	This table is responsible for network address translations. Since IPTables is a stateful packet filter, translations occur on the connections not on the packets. It is important that only the first packet of each connections reaches this table, because one connection can only be NATed once and the selected NAT mapping is used for the rest of the packets in the connection. The NAT table is registered to the PREROUTING, POSTROUTING and to the OUTPUT hooks with a lower priority than the filter table, meaning that it receives the packets right before the filter table.

Note
Although it is possible to drop a packet at the NAT table, it is not recommended, because the filter table is responsible for dropping packets. If a packet is dropped at the NAT table it cannot reach the filter table. If a packet is accepted or NATed at the NAT table it has to be accepted at the filter table as well to pass.

Basically, two types of NATs exist: source address, and destination address. They have various special subtypes such as redirect and masquarade. Both NATs can be configured in this table only. Because of the difference between the two NAT types and their influence on the connections, NATs of source type can be configured at the PREROUTING and at the OUTPUT hooks, while NATs of destination type can be configured at the POSTROUTING hook. With careful design, it is possible to translate source and destination NAT on a single specific connection.

mangle table

This table is responsible for various packet mangling like modifying the TTL of the packet, changing the TOS bits, or setting the FWMARK on the packet. This is the table where the biggest alterations take place in the packet and also where serious administration problems can be experienced.
This table is registered to the PREROUTING, INPUT, FORWARD, OUTPUT, and to the POSTROUTING hooks with the lowest priority meaning that it gets the packets first, right before the other tables.

The default deny or default drop approach is only partially true in this table. As packet filtering occurs at the filter table, the default drop configuration must be utilized only at this table. It is not needed and not possible either, to implement a default drop configuration on all the tables. If a packet targets a local application or leaves the firewall, it has to pass successfully the filter table, so it is enough to drop packets there.

Tip
To build a fully functioning configuration, it is a good approach to accept all the packets on the mangle and on the NAT tables and drop all unnecessary packets on the filter table.

Take special care with NATted packets/connections as they appear with their new addresses after the translation so the NATed address must be accepted on the filter table, for example.

A.3.2.1. Connection tracking

Besides the filter, NAT and mangle tables, another invisible table exists. This table is responsible for connection tracking. IPTables is a stateful packet filter and the statefulness is provided by the connection tracking subsystem which is represented by this table.

This table only checks the relations of the packet towards the connections already investigated. It never drops or rejects any packet, only sets the state information of the packets/connections.

This table is registered to the Prerouting and to the Output hooks with the possible lowest priority, meaning it gets the packets before the mangle table.

A.3.3. Chains

Using the Netfilter hooks the tables provide the functionalities of the packet filter subsystem in Linux. As tables only provide a container for the policy of a specific functionality, some configuration evaluation system is needed. Like many packet filter implementations IPTables uses ACLs for the evaluation mechanism. While other implementations use only one or a limited number of lists, IPTables can have nearly an infinite number of these. The ACLs are called chains in IPTables.

IPTables chains consists of rules which are evaluated from the top (beginning) of the list to the bottom (end). The evaluation stops at the first matching rule if a verdict is set.

Note
It is possible that a rule does not make a verdict on the packet, in that case the evaluation continues.

The evaluation can jump to another chain and later can return to the original one, however if the packet matches on any of the chains the evaluations stops. Each chain resides in a specific table and controls the policy of that given table.

There are basically two types of chains.

- built-in chains
- created chains

Every table contains built-in chains for each of the hooks it has. Every packet that a table on a specific hook gets is put on the specific chain of the given table. The evaluation of this chain is the basis of the verdict of the packet. For example, the filter table has three built-in chains: Input, Forward and Output.

Besides built-in chains, it is possible to create new chains to ease the management of the configuration and can direct the packets on these custom chains by jumping on it.

Note
It is possible to jump to a chain in another table. A smart organization of chains makes the configuration easier to understand and makes the evaluation faster.

Example A.1. Chaining
If a chain has 100 rules, in the worst case 100 rules must be evaluated, but if they are separated into 10 sub-chains — which is usually possible — then even in the worst case 20 rules must be evaluated only.

A.3.4. Rules

The next building blocks of the IPTables configuration are the rules. Tables and chains themselves provide only a container, interface and an evaluation mechanism, but it is the rules that describe the core configuration.

During the evaluation of a chain, actually the rules are evaluated one by one. Every packet is run through this process and the match is checked against each rule in the chains.

The rules consist of two main parts:

- match, and
- target.

Each packet is tested whether that packet and its related status information is matching the match part of the rule. If a match occurs the target part is used.

Note
It is possible that a rule has no target part. In this case nothing happens, only the rules counter is incremented.

If a rule has no match part, all packets match that given rule.

A.3.4.1. Matches

The Netfilter/IPTables system handles matches very flexibly. Almost all aspects of a given packet/connection is possible to be matched. Basically, the matches are just plugins in the framework making it very extendible. Due to the extendibility, a large variety of matches exist.

The match part of a rule can have multiple matches and matching the rule requires that all of the matches are matched. In technical sense, the matches are ANDed together in a rule. If OR-ed matches are required then multiple rules are needed. The most common matches are source/destination address, protocol, source/destination port, TCP flags, connection state (based on the conntrack information), ICMP types and various different mark (FWMARK, CONNMARK) matches. For a full list of matches, see the iptables(8) manpage and the *Appendix C, Further readings (p. 485)*.

A.3.4.2. Targets

Targets define what shall happen with the matched packet. The targets, similarly to the matches are extensions and also behave similarly. Targets can perform two actions:

- modify some parameter of the packet/connections (for example, IP addresses, TOS bits, TTLs, MARKs), and
- provide a verdict.

Note
Not every target has a verdict, meaning that the evaluation of a chain does not end with that matching rule. Remember, the evaluation is ended when a verdict is set, so it is necessary to know which target has a verdict, in other words which one is the final target.

For example, the TTL target, which modifies the TTL of the packet cannot ACCEPT it. For ACCEPT, another rule or a default policy is needed.

Another option for a target is to jump to another chain. In case of jumping to an other chain, the evaluation continues in the new chain. If the other chain is evaluated and no match occurs or no verdict is set, the evaluation continues in the original chain from the next rule after the jump.

The most commonly used targets are ACCEPT, DROP, REJECT, TOS, TTL, [DS]NAT and TPROXY.

A.3.5. Configuration summary

Due to the complexity of the framework, creating configurations with IPTables is a challenging task. For a well-organized and working setup, a very careful design is required. It is very important to use the different tables for their own purposes and to create a working default deny setup.

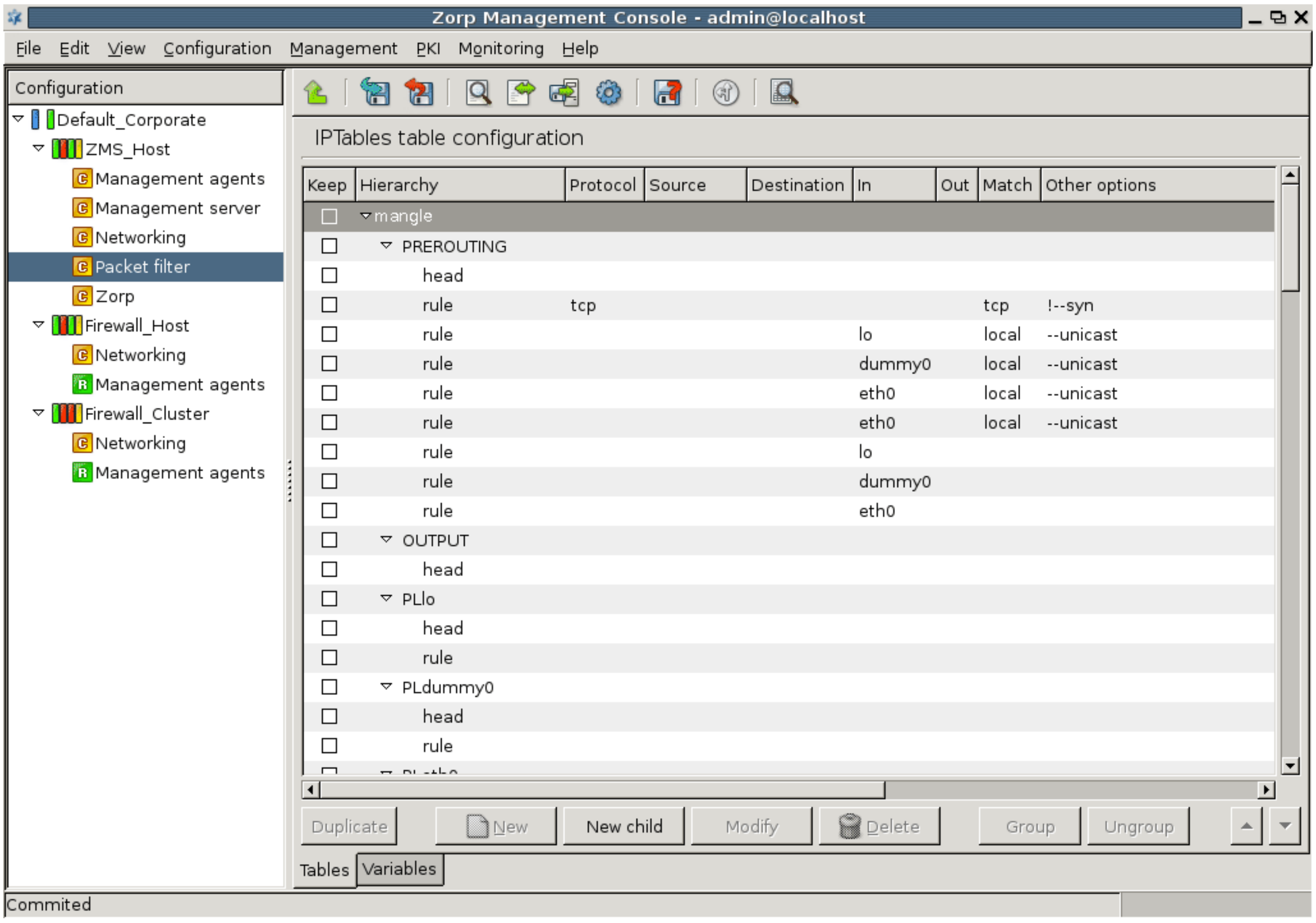
To make this task easier, ZMS automatically generates the packet filtering ruleset, the configuration of which corresponds to the configuration of your Zorp Gateways.

A.4. Managing packet filter rules in ZMC

Note
Packet filtering rules are created and managed automatically by ZMS. Usually it is not required nor recommended to modify them manually. If the transfer of traffic is required without application-level inspection, create a packet filter service (see *Procedure 6.4.1, Creating a new service (p. 115)* for details). To enable access to services running on firewall hosts (e.g., SSH access), see *Section 9.4, Local services on Zorp (p. 232)*.

Typically, the packet filtering rules have to be modified when traffic without terminating it on Zorp has to be forwarded, like forwarding IPSec VPN connections.

ZMC provides an easy and comfortable way to configure the packet filter system on managed hosts. To configure the packet filter the **Packet Filter** component must be added to the managed host. The default configuration defines a default deny/drop setup which means that all packets are dropped and logged in the filter table on the INPUT and FORWARD chains since it is enough to disable all passing traffic on the firewall and there is no need to have drop rules on any other chains. By default, the policy permits all outgoing packets.


Zorp Management Console - admin@localhost
File Edit View Configuration Management PKI Monitoring Help
Duplicate
New
New child
Modify
Delete
Group
Ungroup
Tables
Variables

Configuration
- Default_Corporate
  - ZMS_Host
    - Management agents
    - Management server
    - Networking
    - Packet filter
    - Zorp
  - Firewall_Host
    - Networking
    - Management agents
  - Firewall_Cluster
    - Networking
    - Management agents

IPTables table configuration

Keep	Hierarchy	Protocol	Source	Destination	In	Out	Match	Other options
☐	mangle							
☐	PREROUTING							
☐	head							
☐	rule	tcp					tcp	!--syn
☐	rule				lo		local	--unicast
☐	rule				dummy0		local	--unicast
☐	rule				eth0		local	--unicast
☐	rule				eth0		local	--unicast
☐	rule				lo			
☐	rule				dummy0			
☐	rule				eth0			
☐	OUTPUT							
☐	head							
☐	PLlo							
☐	head							
☐	rule							
☐	PLdummy0							
☐	head							
☐	rule							

Commited

Figure A.3. The Packet filter component

To use the `Packet Filter` component a basic (but preferably higher) understanding of Netfilter/IPTables is required. This component has three basic purposes:

- add/delete/modify rules,
- generate configuration, and
- search/review the ruleset.

Creating all of the packet filter rules for the firewall and keeping the ruleset in synchrony with Zorp is a very challenging and time-consuming task therefore the manual configuration is assisted by a feature which generates the ruleset of the policy. Using this feature of generating the ruleset does not raise any boundaries, neither limits the administration and does not render the configuration inflexible either. The generated policy is just a skeleton which can be modified as any ordinary ruleset.

A.4.1. Configuration management: iptables-utils

The packet filter configuration must be activated on every startup of the firewall or the managed host therefore the configuration is stored in configuration files on the machine. ZMS uses the `iptables-utils` package to handle the packet filter configuration. This package is responsible for loading the ruleset upon startup and making the administration easier by using variables. It also prevents accidental lock-out from the box by a misconfiguration. This package is a self-sufficient program and can be used separately from ZMS.

For managing the configuration, the `iptables-utils` uses four configuration files.

- `Iptables.conf.var`
  This file stores the variables which can be used in the policy. These variables are not the same as ZMS variables and not involved in ZMS-based configuration.

- `Iptables.conf.in`
  This file stores the policy itself with unresolved variables.

- `Iptables.conf.new`
  This file stores the generated configuration from the `iptables.conf.in` and `iptables.con.var` files meaning that the variables are resolved here.

- `Iptables.conf`
  This file stores the actual configuration. The startup policy is loaded from this file.

Three small utilities are used to manage these files and all of them form part of the `iptables-utils` package. To generate the `iptables.conf.new` file `iptables-gen` util is used. To test the new configuration and to prevent lock-outs `iptables-test` is needed, which loads the policy from `iptables.conf.new`, lets it run for 10 seconds and then reloads the old configuration, which is assumed to be functional, from the `iptables.conf` file. If the configuration is working the new configuration can be made effective with the `iptables-commit` util, which loads the new configuration and replaces the `iptables.conf` file.

To control the packet filter system the `/etc/init.d/iptables`-utils is used. It is the `init-script` which also loads the configuration policy upon start-up. When starting the utils, it only loads the policy stored in the `iptables.conf` file. During restart, it generates a new configuration with `iptables-gen` and attempts to load it. In case any error occurs, it reverts back to the old configuration stored in the `iptables.conf` file. If it succeeds, it replaces the `iptables.conf` with the `iptables.conf.new`. For further information on the `iptables-utils`, see the manual page of the utility.

During the ZMS configuration, the `iptables-utils` creates the `iptables.conf.in` and the `iptables.conf.new` files on the managed hosts. Although by default there are no variables used with ZMS, it is possible to use them. To successfully deploy the new configuration, restart the component in order to regenerate the modified and uploaded configuration.

Warning
Without restarting the component, the new configuration is not generated and the modifications are ineffective.

If iptables rules are manually reloded (that is, using the `/etc/init.d/iptables-utils reload` command, or the Packet Filter ZMC component), make sure to reload Zorp as well. Otherwise, the packet filtering services (PFServices) will not function.

A.4.2. Modifying the ruleset

Modifying the ruleset basically means creating new rules/chains, modifying some of their parameters or simply deleting them. This component provides a clean interface for doing these tasks.

Note

Packet filtering rules are created and managed automatically by ZMS. Usually it is not required nor recommended to modify them manually. If the transfer of traffic is required without application-level inspection, create a packet filter service (see *Procedure 6.4.1, Creating a new service (p. 115)* for details). To enable access to services running on firewall hosts (e.g., SSH access), see *Section 9.4, Local services on Zorp (p. 232)*.

Typically, the packet filtering rules have to be modified when traffic without terminating it on Zorp has to be forwarded, like forwarding IPSec VPN connections.

A.4.2.1. Adding new packet filter chains and rules

The packet filter ruleset can be managed on the **Ruleset** tab of the **Packet filter** ZMC component.

The ruleset consists of four basic elements that are organized into a tree. The four elements can be found on different levels of the layout tree.

1. The root elements are the tables which are fixed and cannot be modified in any way.
2. Each table holds a number of table-specific chains. Both types of chains, built-in and user-defined chains, are on the second level of the tree. Built-in chains cannot be deleted and only their default policy can be modified. To add a new chain to a table, select the table and click **New Child**. Alternatively, select an existing chain of the table and click **New**.
   The order of the chains in the table is not important and does not influence the behavior of the ruleset.
3. The child entries of the chains are the rules. To create a new rule in a chain, select the chain and click **New Child**. Alternatively, select an existing rule of the chain and click **New**. For easier overview and management the rules can be grouped together. Groups and rules that do not belong to any group appear on the third level of the tree. To create a group from the rules, select the rules you want to group, right-click on the selected rules, and select **Group** from the local menu.
4. Rules that belong to a group appear on the fourth level of the tree.

Each rule is represented as a row in the table together with its properties (matches and targets) in the columns. Unlike chains, the order of the rules is important. The order can be changed with the small triangle buttons on the right. To create a rule the matches and the targets need to be configured. To modify a rule, double click the rule and change the match and target part. The most commonly used matches and the targets can be set on the **General options** tab, while other rarely used matches can be configured on the **Advanced options** tab.

For further information on the matches and targets, see the iptables(8) manpage and the *Appendix C, Further readings (p. 485)*.

Tip

The direction of a rule can be changed by selecting **Swap directions** from the local menu.

A.4.3. Understanding the packet filter ruleset

Packet filter policies can be organized in different ways. You are free to create your own arrangements. The skeleton itself provides only one proved and tested way. Although the skeleton is not necessarily full and ready for use, it can be extended and finetuned to meet the requirements. Of course, there can be situations when the skeleton policy is ready for use without modifications.

In a skeleton-based policy, the generated and user-defined rules are mixed and can function together. Without using skeletons only user-defined rules exist. Generated rules are either based on information collected from other components, such as `Networking`, `IPSec VPN` or `Zorp`, or set during the generation of the skeleton.

Tip
It is recommended to configure packet filter function through skeletons.

Note
UDP port 4500 is automatically opened if the *`Nat Traversal`* option is enabled in the **VPN** component.

The ruleset is generated automatically when the configuration of Zorp is modified in a way which affects the packet filter configuration. User-defined rules remain untouched if specified as a keep rule, otherwise the generator removes these rules from the ruleset.

The generator automatically collects the information required to generate the ruleset from the different ZMC components. After the generator finished, the newly created configuration is presented and can be used the same way as an ordinary user-created configuration.

The new configuration contains both generated and user defined rules.

Note
It is recommended to keep all user-defined rules and regenerate the configuration every time relevant modifications are made.

Since the skeleton consists of generated and user created rules, the relative order of their relations is very important. The generator creates and modifies chains and rules as well. In every chain which is modified by the generator, the generated rules always come first and user-created rules follow them in the framework. The generation of rules drops those user-created rules which are not marked with 'keep rules'. Every other chains are left intact.

The order of the generated rules is not specified, but all of them are placed before the user-created ones. The relative order of the user-created rules is kept during the rule generation processes, although all the user-created rules are moved after the generated ones. There are two exceptions: the head group and the catch-all rules.

Every chain modified/touched by the generator has a head group. The role of the head group is to provide a way to insert user-created rules before the generated ones. The head group is a generated group and is placed as the first entry of the chain so every rule which is in the head group of the specific chain precedes all of the generated rules in the chain. The rules in the head group remain in place during the skeleton generation.

Some chains (built-in chains and user-defined ones from the IPTables point of view) have a catch-all rule(s). These rules try to provide a default policy for those chains which need it, for example the chains in the filter table.

The purpose of the catch-all rules is a default policy so they must be at the end of each chain following the user-created rules. No user created rule can follow them. Any rule appended after these catch-all ones are moved forward during the generation.

For the filter table chains the catch-all rules consist of a LOG and a DROP rules to log the packets before they are dropped. The LOG rule also logs the chain to make it possible to trace where the packet was dropped. These two rules, but especially the DROP rule provides the default drop approach from the packet filter point of view for the host.

The catch-all rule in the mangle table only ACCEPTs the packets. ACCEPTing any packet in any table other than the filter does not necessarily mean letting it pass or allowing it to the application. Accepting packets in the mangle table means that neither the packet nor its associated mark values are modified; the packet is only passed for further processing.

After skeleton generation, the modified chain looks like the following:

1. Head group at the beginning with the user created rules in preserved order.
2. Generated rules with an unspecified order.
3. User-created rules, again, with also a preserved order.
4. Catch-all rules

A.4.3.1. Marking packets for transparent proxying

This section describes how Zorp selects and marks the packets that will be transparently proxied.

Note
In Zorp version 3.3FR1 and earlier, this functionality was achieved using the tproxy table. Starting with Zorp version 3.4, the tproxy table is not available in Zorp.

As of Zorp version 3.4, the Zorp Gateway sets a mark on incoming packets that are to be transparently proxied. Every packet having this specific mark is sent to the LO interface using a policy routing rule. Zorp uses the highest bit for marking the packets, that is, *0x80000000*. In the iptables ruleset of Zorp, this means the following rule: *-A LOeth0 --match mark --mark 0x80000000/0x80000000 --jump ACCEPT*

Therefore, in case you use any marks, make sure that:

- you do not use the highest bit for marking; and that
- you use the following mask when setting or checking your marks: */0x7fffffff*

When using your own iptables rules to mark packets for transparent proxying, the marks must set the highest bit as well, and set the proper mask, for example, using the *--tproxy-mark 0x80000000/0x80000000* iptables option.

Marking the packets is responsible only for redirecting the packets for transparent proxying, no filtering is performed. The marked packets and connections must be ACCEPTed in the filter table as well to make the packet pass and reach their destination which is a dispatcher for transparently proxied packets.

Note
By default, Zorp directs every packet to the kzorp target in the mangle table. Explicitly accepting packets earlier in the iptables rules provides a way to avoid the kzorp kernel module.

To ensure that the selected packets do not reach the kzorp target, add rules to accept the packets to every chain of the mangle table (PREROUTING, FORWARD, POSTROUTING) before the following rule: `-A <name-of-the-chain> --jump KZORP --tproxy-mark 0x80000000/0x80000000`. You can use zone matches as well to select the packets to accept.

A.4.3.2. The filter table

The filter table is the place where the real filtering takes place so it is very important to understand how and why packets/connections are ACCEPTed or DROPed here.

Every connection that the firewall has to allow must be ACCEPTed here in some way. Basically, three kinds of traffic is handled at the filter table.

- incoming
- forwarded
- outgoing

The generated ruleset allows every outgoing packet, (reply packets are allowed as well), so that OUTPUT chain ACCEPTs every packet. As Zorp is an application level solution no packet forwarding takes place, therefore the FORWARD chain DROPs the packets.

This chain has a head group, like all generated chains, which is responsible for spoof protection. See also section *Spoof protection*. It has the LOG+DROP catch-all rules at the end as well.

Note
It is recommended not to manually configure the packet filter to allow packet forwarding, though it is possible.

The incoming traffic is basically handled in the INPUT chain and its subchains since the filter table is only meant for filtering (DROPping REJECTing and ACCEPTing) the packets and the connections. As every ruleset generated chain, the INPUT also begins with the head group, just as it works everywhere else.

After the head group, the evaluation continues with *`spoof`* protection, followed by the *`broadcast`* chain and *`noise`* filtering. Noise filtering is responsible for DROPping every network traffic/packet that is considered junk like broadcast messages or anything else that you want to drop without LOGing it. This filtering takes place in the noise chain which is generated by the ruleset generator. The noise chain begins with a head group and is followed by DROP rules dropping (but not logging) packets going to the broadcast address of the network interfaces. As a catch-all rule, the chain ends with a RETURN rule which jumps back to the INPUT chain and the ruleset evaluation continues from there.

Tip
It is recommended to finetune the noise chain because the requirements and the network junk vary from site to site.

If you want to DROP something, add the appropriate DROP rules to the noise chain, but be careful not to DROP anything which is not a junk and should be LOGed later.

Remember not to put any rule ACCEPTing any traffic as this rule would ACCEPT it for the filter table. Use RETURN instead as the noise chain is only for DROPping the junk and other rules would interfere with the configuration.

After that, various ICMP packets are ACCEPTed which are RELATED to already established connections. Not every type of ICMP packets are allowed, only those that are needed for the correct and secure operation of the firewall. These are the following:

- destination-unreachable(3)
- source-quench(4)
- time-exeeded(11)
- parameter-problem(12)

ICMP packet types are sent as responses for an already established connection by the remote peer indicating some network problem. RELATED means here that only those messages are ACCEPTed which are responses for a known connection, messages related to an unknown connection are not ACCEPTed. For further information on RELATED and ICMP types, see *Appendix C, Further readings (p. 485)*.

After the ICMP rules, the next rule ACCEPTs every packet which belongs to an already ESTABLISHED connection. This way the reply packets for the outgoing connections are ACCEPTed. To track the state and the relation of a packet to the known connections the conntrack table is used. This table assigns a state information to every packet which is checked here.

The *`defaults`* group is next, which is responsible for keeping some necessary default rules. There is no need to modify these default rules although it is possible to add other rules. The first rule ACCEPTs every packet that has been marked for transparent proxying. Transparent proxying packets/connections in the mangle table is just redirecting them to the dispatchers (so they can handle transparent traffic), but every packet must be ACCEPTed on the filter table as well, so transparently proxying them on the mangle table is insufficient in this situation. When a connection is marked for transparent proxying, every packet belonging to that connection receives the same *`tproxy-mark`*. This way every previously TPROXYed packet can be ACCEPTed with only one rule. Another use of this mark is to flag those packets/connections which are implicitly requested by Zorp and ACCEPT them as well with this rule. These connections are usually the secondary connections of traffic proxied by Zorp, for example the data channel of an FTP session. When Zorp listens for such a connection, it implicitly requests transparent proxying and flagging so there is no need to manually add rules for them — ZMS automatically generates the required rule into the PREROUTING chain of the mangle table: *`-A PREROUTING --match socket --transparent --jump MARK --set-mark 0x80000000/0x80000000`*. Not that it is also needed to allow/ACCEPT these connections on the filter table and this tproxy matching rule realizes that as well.

Altogether, in the defaults group every packet is ACCEPTed which has already been handled (for example, transparently handled packets) or is related to, or is part of already known connections. Every other packet must be filtered.

The next group is called local_services which is responsible for handling the packet filter configuration of the local or native services. These services can be set as the first step of the ruleset generation. For every service and for every selected zone an ACCEPT rule is generated matching the protocol, the port and the source zone addresses.

Following the local_services group, the routing group can be found. Like in the tproxy table, the rules are divided to subchains based on the incoming interface of the packets. The routing group is the placeholder for such rules that separate the traffic. For each and every interface a LO<interface name> chain exists and the rules in the routing group jump the evaluation to these chains according to the incoming interface and the IP addresses (primary and alias addresses) of the interface pairs. Every further production rule is located in the

LO* subchains. Every LO* chain holds the rules that apply to the traffic coming in on the appropriate interface and target the host itself.

Note
The target IP address of the traffic is not necessarily the IP address of the interface, it can be the IP address of any of its alias interfaces.

The LO* chains begin with the usual head group. After the head group, the rules generated by the ruleset can be found, but the order of the rules is not specified. The information for the rules come from two components: the `Zorp` and the `VPN`.

For every VPN connection, two rules are generated: one for ACCEPTing UDP packets from the remote end of the connection on source and destination port 500 and another rule ACCEPTing ESP (protocol: 50) packets from the remote end of the connection. *These rules only allow the encrypted communication everything inside the tunnel, the real payload which comes on the ipsec* interfaces must be filtered separately like any other traffic.*

The other rules placed in the LO* chains are generated based on the information from the Zorp components. For every non-Transparent dispatcher in Zorp an ACCEPT rule is created in one of the LO* chains according to the Address setting of the dispatcher. The configured Address specifies into which sub LO* chain should the rule be placed in.

Each rule ACCEPTs packets to destination ports configured in the Proxy port attribute where the dispatcher listens for connections.

At the end of the INPUT chain, every packet is LOGged and DROPed. Although normally no packet can get to this point of evaluation, because they are thrown to the LO* subchains and at the end of the subchains everything is LOGed and DROPed again. The LOG rule also logs the place (“filter/INPUT”) and the verdict (“DROP”) in the logs.

A.4.3.3. Spoof protection

The purpose of spoof protection is to ensure and enforce that the source address of any packet is in the network which is connected to and reachable through the interface the packet is coming from and only from and through that interface.

Example A.2. Protection against spoof
If a packet comes from the intranet, the source address of the packet must be in the address range of the intranet. If the source address of the packet is in the range of intranet, the packet can only come in on the interface connected to the intranet and from no other interfaces. The last rule is especially important for the internet case where the internet address range (0.0.0.0/0) matches any address, but it must be ensured that the source address does not belong to any other network of the firewall. As the access control of the firewall is based on IP addresses it is very important to be protected against IP spoof attacks.

However, network topologies can be very complex which renders the spoof protect ruleset very complicated. The ruleset generator automates the generation of these types of rules as well.

To successfully generate a secure ruleset, the generator must be aware of the network topology of the neighbourhood of the firewall. In the `Networking` component for every interface you have to configure the connected (Connects) zones and set the **Spoof protection** also.

The connected zones set which zones, which network addresses are reachable through that interface. The core of the spoof protection is the spoof chain which is in the filter table. The spoof chain is jumped from the INPUT and FORWARD chains as both incoming and forwarded connections must be checked against spoof attacks. The chain begins with the crucial head group followed by the loopback interface handling.

First, every packet is ACCEPTed that comes in on the LO interface, then everything is LOGed (with a 'filter/spoof DROP' prefix) and DROPed that has source address in the 127.0.0.0/8 network, but the incoming interface is not LO. Spoof check is done on a per-interface basis. Since all the interfaces are checked separately, an SP<interface name> chain is created for each interface.

After the loopback interface check taking place in the spoof chain according to the incoming interface, the evaluation is jumped to the appropriate SP* chain.

Note
Alias interfaces work and behave just like their master interface, so anything configured for an alias interface applies to its master interface as well.

In the SP* chains, spoof attacks are checked based on what was configured in the Network component. Every SP* chains have almost the same ruleset, the rules differ only in their target part. The chains contain one or two rules (depending on whether it is a LOG + DROP or a RETURN rule) for every addresses of zones that are connected to the given interface. Every rule has a network address of a zone as a source match.

The order of the rules are very important. The rules are sorted by the netmask of the source match in descending order. This way the rules with the highest netmask come first, which also means that the smallest networks are matched first. It is inevitable to have this order to match the tightest network first, otherwise the tighter rules would never match, because the bigger networks would match previously.

The target of a rule depends on whether the zone containing the address of the match is selected as a connected zone for the interface to which the chain applies. If it is selected, the target is RETURN otherwise there are two rules: a LOG and a DROP one.

This ruleset ensures that if a zone is connected to an interface, the source address of packets coming in on that interface must match any networks of the zone. It also ensures that if the zone, which contains the network the packet is sourced from, is connected to any other interface then the packet is not allowed, it would be LOGged and DROPped.

Zones that have no addresses associated with or not connected to any interface are not used and ignored.

Note
To have a full and secure spoof protection ruleset it is very important to set the connected zones, otherwise the spoof protect is incomplete. Do not unset the Spoof protection flag in the `Networking` component.

A.4.4. The Rule Search window

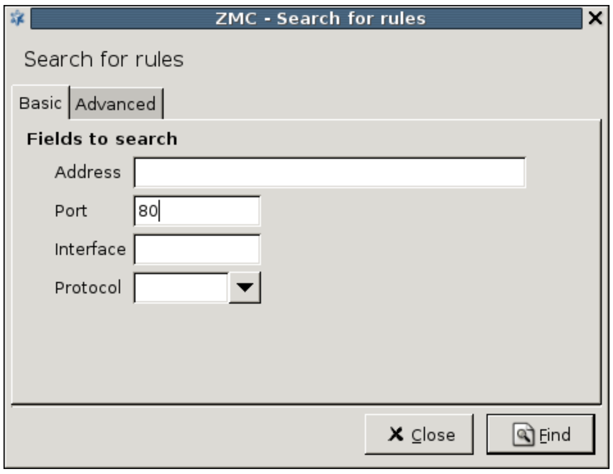
Adding new rules and generating the ruleset is only one part of the configuration process. During the lifecycle of the firewall you have to review and modify the rules. The `Packet Filter` component provides a Rule Search window as a tool for searching the ruleset.

The Rule Search window is accessible from the button bar.

A.4.4.1. Procedure – Using Rule Search

Step 1. Click on the **Rule search** icon.
A new window appears.

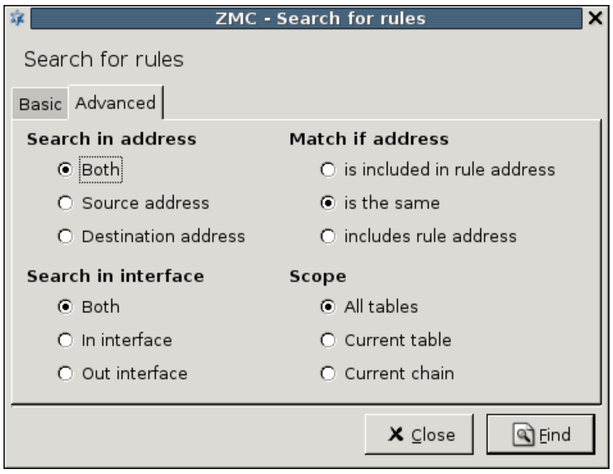
Step 2. Fill in the search fields of the **Basic** tab and click **Find**.
The matching rules are selected from the background in the `Packet Filter` component window.



Address
Port 80
Interface
Protocol

Figure A.4. Using Rule search

Step 3. Configure search-criteria and field interpretation on the **Advanced** tab.



Search for rules

Basic | Advanced

Search in address

- [✓ Both]
- Source address
- Destination address

Match if address

- is included in rule address
- [✓ is the same]
- includes rule address

Search in interface

- [✓ Both]
- In interface
- Out interface

Scope

- [✓ All tables]
- Current table
- Current chain

Figure A.5. Advanced Rule search

Tip

If a search criteria does not match the requested rule(s), check the settings on the **Advanced** tab.

Appendix B. Keyboard shortcuts in Zorp Management Console

B.1. Function keys

`Ctrl-F1`	Display the tooltip of the selected item.
`Shift-F1`	Display the help of the selected item.
`F5`	Refresh the screen and the status of the displayed objects.
`F8`	Activate the splitter bar. Use the cursor keys to resize the selected panel.
`F9`	Hide or display the configuration tree.
`F10`	Activate the main menu.
`Shift-F10`	Display the local menu of the selected item.
`Tab, Shift-Tab`	Move keyboard focus to the next or the previous item.
`Ctrl-Tab, Ctrl-Shift-Tab`	Move keyboard focus to the next or the previous item if Tab has function in the window (for example, in the Text Editor component).

B.2. Shortcuts

`Ctrl-S`	Commit (save) the configuration.
`Ctrl-Shift-E`	View the configuration stored in ZMS.
`Ctrl-E`	Compare the configuration stored in ZMS with the one, running on the selected host.
`Ctrl-U`	Upload the configuration to ZMS.
`Ctrl-I`	Display the Control Service dialog of the selected component.
`Ctrl-X`	Move the selected object to the clipboard.
`Ctrl-C`	Copy the selected object to the clipboard.
`Ctrl-V`	Paste the object from the clipboard.
`Ctrl-F`	Find a keyword in the active list or tree.
`Ctrl-A`	Select all objects of the active list or tree.
`Shift-+`	Open all objects of the active list or tree.
`Shift--`	Close all objects of the active list or tree.
`Shift-+`	Open all objects of the active list or tree.
`Shift--`	Close all objects of the active list or tree.
`Ctrl-Q`	Close ZMC.
`Esc`	Close the active dialog.

B.3. Access keys

Every button, menu item, checkbox, and so on in ZMC has an access key — an underlined letter in the name of the object. Pressing `Alt-<accesskey>` activates the object. For example, it selects or unselects the checkbox, or activates the button. For example, `Alt-R` renames an interface on the **Networking** component.

Appendix C. Further readings

The following is a list of recommended readings concerning various parts of Zorp administration.

Note
Note that URLs can change over time. The URLs of the online references were valid at the time of writing.

C.1. Zorp-related material

- Guides, manuals, and tutorials for Zorp are available at *https://docs.balasys.hu/*

C.2. General, Linux-related materials

- *The Linux Documentation Project*
- *Linux Advanced Routing and Traffic Control*

C.3. Postfix documentation

- Author's name. Title. Place of publication: publisher, year. *The Postfix Home Page*
- Blum, Richard. Postfix. SAMS Publishing, 2001. ISBN: 0672321149
- Dent, Kyle D. Postfix: The Definitive Guide. O'Reilly Associates, 2004. ISBN: 0596002122

C.4. BIND Documentation

- *BIND Manual Pages*
- Albitz, Paul, and Liu, Cricket. DNS and BIND. O'Reilly Associates, 2001. ISBN: 0596001584

C.5. NTP references

- *NTP Documentation*
- *NTP Documentation* RFC 1305, Network Time Protocol Specification, Implementation and Analysis

C.6. SSH resources

- *Official site of the OpenSSH project*
- Barrett, Daniel J. Ph.D., and Silverman, Richard E. SSH: The Secure Shell The Definitive Guide. O'Reilly Associates, 2001. ISBN: 0596000111

C.7. TCP/IP Networking

- Stevens, W., and Wright, Gary. TCP/IP Illustrated: Volumes 1-3. Addison-Wesley, 2001. ISBN: 0201776316

- Mann, Scott. Linux TCP/IP Network Administration. Prentice Hall, 2002. ISBN: 0130322202

C.8. Netfilter/IPTables

- *Official site of the Netfilter project*

C.9. General security-related resources

- Garfinkel, Simson, et al. Practical UNIX and Internet Security, 3/E. O'Reilly Associates, 2003. ISBN: 0596003234

C.10. syslog-ng references

- The syslog-ng Administrator Guide
  https://docs.balasys.hu/

C.11. Python references

- *Official Python documentation page*

C.12. Public key infrastructure (PKI)

- *RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*
- *RFC 6960, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*

C.13. Virtual Private Networks (VPN)

- Wouters, Paul, and Bantoft, Ken. Openswan: Building and Integrating Virtual Private Networks. Packt Publishing, 2006. ISBN 1904811256.
- Feilner, Markus. OpenVPN: Building and Integrating Virtual Private Networks, 2006. Packt Publishing. ISBN 190481185X.

Appendix D. Zorp Professional End-User License Agreement

D.1. 1. SUBJECT OF THE LICENSE CONTRACT

1.1 This License Contract is entered into by and between Balasys and Licensee and sets out the terms and conditions under which Licensee and/or Licensee's Authorized Subsidiaries may use the Zorp Professional under this License Contract.

D.2. 2. DEFINITIONS

In this License Contract, the following words shall have the following meanings:

2.1 Balasys

Company name: Balasys IT Ltd.

Registered office: H-1117 Budapest, Alíz Str. 4.

Company registration number: 01-09-687127

Tax number: HU11996468-2-43

2.2. Words and expressions

Annexed Software

Any third party software that is a not a Balasys Product contained in the install media of the Balasys Product.

Authorized Subsidiary

Any subsidiary organization: (i) in which Licensee possesses more than fifty percent (50%) of the voting power and (ii) which is located within the Territory.

Balasys Product

Any software, hardware or service licensed, sold, or provided by Balasys including any installation, education, support and warranty services, with the exception of the Annexed Software.

License Contract

The present Zorp Professional License Contract.

Product Documentation

Any documentation referring to the Zorp Professional or any module thereof, with special regard to the reference guide, the administration guide, the product description, the installation guide, user guides and manuals.

Protected Hosts

Host computers located in the zones protected by Zorp Professional, that means any computer bounded to network and capable to establish IP connections through the firewall.

Protected Objects

The entire Zorp Professional including all of its modules, all the related Product Documentation; the source code, the structure of the databases, all registered information reflecting the structure of the Zorp Professional and all the adaptation and copies of the Protected Objects that presently exist or that are to be developed in the future, or any product falling under the copyright of Balasys.

Zorp Professional

Application software Balasys Product designed for securing computer networks as defined by the Product Description.

Warranty Period

The period of twelve (12) months from the date of delivery of the Zorp Professional to Licensee.

Territory

The countries or areas specified above in respect of which Licensee shall be entitled to install and/or use Zorp Professional.

Take Over Protocol

The document signed by the parties which contains

a) identification data of Licensee;

b) ordered options of Zorp Professional, number of Protected Hosts and designation of licensed modules thereof;

c) designation of the Territory;

d) declaration of the parties on accepting the terms and conditions of this License Contract; and

e) declaration of Licensee that is in receipt of the install media.

D.3. 3. LICENSE GRANTS AND RESTRICTIONS

3.1. For the Zorp Professional licensed under this License Contract, Balasys grants to Licensee a non-exclusive,

non-transferable, perpetual license to use such Balasys Product under the terms and conditions of this License Contract and the applicable Take Over Protocol.

3.2. Licensee shall use the Zorp Professional in the in the configuration and in the quantities specified in the Take Over Protocol within the Territory.

3.3. On the install media all modules of the Zorp Professional will be presented, however, Licensee shall not be entitled to use any module which was not licensed to it. Access rights to modules and IP connections are controlled by an "electronic key" accompanying the Zorp Professional.

3.4. Licensee shall be entitled to make one back-up copy of the install media containing the Zorp Professional.

3.5. Licensee shall make available the Protected Objects at its disposal solely to its own employees and those of the Authorized Subsidiaries.

3.6. Licensee shall take all reasonable steps to protect Balasys's rights with respect to the Protected Objects with special regard and care to protecting it from any unauthorized access.

3.7. Licensee shall, in 5 working days, properly answer the queries of Balasys referring to the actual usage conditions of the

Zorp Professional, that may differ or allegedly differs from the license conditions.

3.8. Licensee shall not modify the Zorp Professional in any way, with special regard to the functions inspecting the usage of the software. Licensee shall install the code permitting the usage of the Zorp Professional according to the provisions defined for it by Balasys. Licensee may not modify or cancel such codes. Configuration settings of the Zorp Professional in accordance with the possibilities offered by the system shall not be construed as modification of the software.

3.9. Licensee shall only be entitled to analize the structure of the Balasys Products (decompilation or reverse-engineering) if concurrent operation with a software developed by a third party is necessary, and upon request to supply the information required for concurrent operation Balasys does not provide such information within 60 days from the receipt of such a request. These user actions are limited to parts of the Balasys Product which are necessary for concurrent operation.

3.10. Any information obtained as a result of applying the previous Section

(i) cannot be used for purposes other than concurrent operation with the Balasys Product;

(ii) cannot be disclosed to third parties unless it is necessary for concurrent operation with the Balasys Product;

(iii) cannot be used for the development, production or distribution of a different software which is similar to the BalaSys Product

in its form of expression, or for any other act violating copyright.

3.11. For any Annexed Software contained by the same install media as the Balasys Product, the terms and conditions defined by its copyright owner shall be properly applied. Balasys does not grant any license rights to any Annexed Software.

3.12. Any usage of the Zorp Professional exceeding the limits and restrictions defined in this License Contract shall qualify as material breach of the License Contract.

3.13. The Number of Protected Hosts shall not exceed the amount defined in the Take Over Protocol.

3.14. Licensee shall have the right to obtain and use content updates only if Licensee concludes a maintenance contract that includes such content updates, or if Licensee has otherwise separately acquired the right to obtain and use such content updates. This License Contract does not otherwise permit Licensee to obtain and use content updates.

D.4. 4. SUBSIDIARIES

4.1 Authorized Subsidiaries may also utilize the services of the Zorp Professional under the terms and conditions of this License Contract. Any Authorized Subsidiary utilising any service of the Zorp Professional will be deemed to have accepted the terms and conditions of this License Contract.

D.5. 5. INTELLECTUAL PROPERTY RIGHTS

5.1. Licensee agrees that Balasys owns all rights, titles, and interests related to the Zorp Professional and all of Balasys's patents, trademarks, trade names, inventions, copyrights, know-how, and trade secrets relating to the design, manufacture, operation or service of the Balasys Products.

5.2. The use by Licensee of any of these intellectual property rights is authorized only for the purposes set forth herein, and upon termination of this License Contract for any reason, such authorization shall cease.

5.3. The Balasys Products are licensed only for internal business purposes in every case, under the condition that such license does not convey any license, expressly or by implication, to manufacture, duplicate or otherwise copy or reproduce any of the Balasys Products.

No other rights than expressly stated herein are granted to Licensee.

5.4. Licensee will take appropriate steps with its Authorized Subsidiaries, as Balasys may request, to inform them of and assure compliance with the restrictions contained in the License Contract.

D.6. 6. TRADE MARKS

6.1. Balasys hereby grants to Licensee the non-exclusive right to use the trade marks of the Balasys Products in the Territory in accordance with the terms and for the duration of this License Contract.

6.2. Balasys makes no representation or warranty as to the validity or enforceability of the trade marks, nor as to whether these infringe any intellectual property rights of third parties in the Territory.

D.7. 7. NEGLIGENT INFRINGEMENT

7.1. In case of negligent infringement of Balasys's rights with respect to the Zorp Professional, committed by violating the restrictions and limitations defined by this License Contract, Licensee shall pay liquidated damages to Balasys. The amount of the liquidated damages shall be twice as much as the price of the Balasys Product concerned, on Balasys's current Price List.

D.8. 8. INTELLECTUAL PROPERTY INDEMNIFICATION

8.1. Balasys shall pay all damages, costs and reasonable attorney's fees awarded against Licensee in connection with any claim brought against Licensee to the extent that such claim is based on a claim that Licensee's authorized use of the Balasys Product infringes a patent, copyright, trademark or trade secret. Licensee shall notify Balasys in writing of any such claim as soon as Licensee learns of it and shall cooperate fully with Balasys in connection with the defense of that claim. Balasys shall have sole control of that defense (including without limitation the right to settle the claim).

8.2. If Licensee is prohibited from using any Balasys Product due to an infringement claim, or if Balasys believes that any Balasys Product is likely to become the subject of an infringement claim, Balasys shall at its sole option, either: (i) obtain the right for Licensee to continue to use such Balasys Product, (ii) replace or modify the Balasys Product so as to make such Balasys Product non-infringing and substantially comparable in functionality or (iii) refund to Licensee the amount paid for such infringing Balasys Product and provide a pro-rated refund of any unused, prepaid maintenance fees paid by Licensee, in exchange for Licensee's return of such Balasys Product to Balasys.

8.3. Notwithstanding the above, Balasys will have no liability for any infringement claim to the extent that it is based upon:

(i) modification of the Balasys Product other than by Balasys,

(ii) use of the Balasys Product in combination with any product not specifically authorized by Balasys to be combined with the Balasys Product or

(iii) use of the Balasys Product in an unauthorized manner for which it was not designed.

D.9. 9. LICENSE FEE

9.1. The number of the Protected Hosts (including the server as one host), the configuration and the modules licensed shall serve as the calculation base of the license fee.

9.2. Licensee acknowlegdes that payment of the license fees is a condition of lawful usage.

9.3. License fees do not contain any installation or post charges.

D.10. 10. WARRANTIES

10.1. Balasys warrants that during the Warranty Period, the optical media upon which the Balasys Product is recorded will not be defective under normal use. Balasys will replace any defective media returned to it, accompanied by a dated proof of purchase, within the Warranty Period at no charge to Licensee. Upon receipt of the allegedly defective Balasys Product, Balasys will at its option, deliver a replacement Balasys Product or Balasys's current equivalent to Licensee at no additional cost. Balasys will bear the delivery charges to Licensee for the replacement Product.

10.2. In case of installation by Balasys, Balasys warrants that during the Warranty Period, the Zorp Professional, under normal use in the operating environment defined by Balasys, and without unauthorized modification, will perform in substantial compliance with the Product Documentation accompanying the Balasys Product, when used on that hardware for which it was installed, in compliance with the provisions of the user manuals and the recommendations of Balasys. The date of the notification sent to Balasys shall qualify as the date of the failure. Licensee shall do its best to mitigate the consequences of that failure. If, during the Warranty Period, the Balasys Product fails to comply with this warranty, and such failure is reported by Licensee to Balasys within the Warranty Period, Balasys's sole obligation and liability for breach of this warranty is, at Balasys's sole option, either:

(i) to correct such failure,

(ii) to replace the defective Balasys Product or

(iii) to refund the license fees paid by Licensee for the applicable Balasys Product.

D.11. 11. DISCLAIMER OF WARRANTIES

11.1. EXCEPT AS SET OUT IN THIS LICENSE CONTRACT, BALASYS MAKES NO WARRANTIES OF ANY KIND WITH RESPECT TO THE Zorp Professional. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, BALASYS EXCLUDES ANY OTHER WARRANTIES, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF SATISFACTORY QUALITY, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS.

D.12. 12. LIMITATION OF LIABILITY

12.1. SOME STATES AND COUNTRIES, INCLUDING MEMBER COUNTRIES OF THE EUROPEAN UNION, DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES AND, THEREFORE, THE FOLLOWING LIMITATION OR EXCLUSION MAY NOT APPLY TO THIS LICENSE CONTRACT IN THOSE STATES AND COUNTRIES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW AND REGARDLESS OF WHETHER ANY REMEDY SET OUT IN THIS LICENSE CONTRACT FAILS OF ITS ESSENTIAL PURPOSE, IN NO EVENT SHALL BALASYS BE LIABLE TO LICENSEE FOR ANY SPECIAL, CONSEQUENTIAL, INDIRECT OR SIMILAR DAMAGES OR LOST PROFITS OR LOST DATA ARISING OUT OF THE USE OR INABILITY TO USE THE Zorp Professional EVEN IF BALASYS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

12.2. IN NO CASE SHALL BALASYS'S TOTAL LIABILITY UNDER THIS LICENSE CONTRACT EXCEED THE FEES PAID BY LICENSEE FOR THE Zorp Professional LICENSED UNDER THIS LICENSE CONTRACT.

D.13. 13.DURATION AND TERMINATION

13.1. This License Contract shall come into effect on the date of signature of the Take Over Protocol by the duly authorized

representatives of the parties.

13.2. Licensee may terminate the License Contract at any time by written notice sent to Balasys and by simultaneously destroying all copies of the Zorp Professional licensed under this License Contract.

13.3. Balasys may terminate this License Contract with immediate effect by written notice to Licensee, if Licensee is in material or persistent breach of the License Contract and either that breach is incapable of remedy or Licensee shall have failed to remedy that breach within 30 days after receiving written notice requiring it to remedy that breach.

D.14. 14. AMENDMENTS

14.1. Save as expressly provided in this License Contract, no amendment or variation of this License Contract shall be effective unless in writing and signed by a duly authorised representative of the parties to it.

D.15. 15. WAIVER

15.1. The failure of a party to exercise or enforce any right under this License Contract shall not be deemed to be a waiver of that right nor operate to bar the exercise or enforcement of it at any time or times thereafter.

D.16. 16. SEVERABILITY

16.1. If any part of this License Contract becomes invalid, illegal or unenforceable, the parties shall in such an event negotiate in good faith in order to agree on the terms of a mutually satisfactory provision to be substituted for the invalid, illegal or unenforceable

provision which as nearly as possible validly gives effect to their intentions as expressed in this License Contract.

D.17. 17. NOTICES

17.1. Any notice required to be given pursuant to this License Contract shall be in writing and shall be given by delivering the notice by hand, or by sending the same by prepaid first class post (airmail if to an address outside the country of posting) to the address of the relevant party set out in this License Contract or such other address as either party notifies to the other from time to time. Any notice given according to the above procedure shall be deemed to have been given at the time of delivery (if delivered by hand) and when received (if sent by post).

D.18. 18. MISCELLANEOUS

18.1. Headings are for convenience only and shall be ignored in interpreting this License Contract.

18.2. This License Contract and the rights granted in this License Contract may not be assigned, sublicensed or otherwise transferred in whole or in part by Licensee without Balasys's prior written consent. This consent shall not be unreasonably withheld or delayed.

18.3. An independent third party auditor, reasonably acceptable to Balasys and Licensee, may upon reasonable notice to Licensee and during normal business hours, but not more often than once each year, inspect Licensee's relevant records in order to confirm that usage of the Zorp Professional complies with the terms and conditions of this License Contract. Balasys shall bear the costs of such audit. All audits shall be subject to the reasonable safety and security policies and procedures of Licensee.

18.4. This License Contract constitutes the entire agreement between the parties with regard to the subject matter hereof. Any modification of this License Contract must be in writing and signed by both parties.

Appendix E. Creative Commons Attribution Non-commercial No Derivatives (by-nc-nd) License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED. BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. *Definitions*

   a. "Adaptation" means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

   b. "Collection" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.

   c. "Distribute" means to make available to the public the original and copies of the Work through sale or other transfer of ownership.

   d. "Licensor" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

   e. "Original Author" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

f. "Work" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

g. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

h. "Publicly Perform" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

i. "Reproduce" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. *Fair Dealing Rights.* Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. *License Grant.* Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections; and,

   b. to Distribute and Publicly Perform the Work including as incorporated in Collections.

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Adaptations. Subject to 8(f), all rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Section 4(d).

4. *Restrictions.* The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested.

b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

c. If You Distribute, or Publicly Perform the Work or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (for example a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Collection, at a minimum such credit will appear, if a credit for all contributing authors of Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

d. For the avoidance of doubt:

   i. Non-waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

   ii. Waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,

iii. Voluntary License Schemes. The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b).

e. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation.

5. *Representations, Warranties and Disclaimer* UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. *Limitation on Liability.* EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. *Termination*

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. *Miscellaneous*

   a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further