



## Administrators Guide

---

### **Clavister® CorePlus™ Version 8.80**

Clavister AB  
Torggatan 10  
SE-891 28 Örnsköldsvik  
SWEDEN  
Phone: +46-660-299200  
Fax: +46-660-12250  
[www.clavister.com](http://www.clavister.com)

Build: 880  
Published 2006-12-19  
Copyright © 2006 Clavister AB.

---

# **Administrators Guide**

## **Clavister® CorePlus™**

### **Version 8.80**

Published 2006-12-19  
Build: 880

Copyright © 2006 Clavister AB.

#### **Copyright Notice**

This publication, including all photographs, illustrations and software, is protected under international copyright laws, with all rights reserved. Neither this manual, nor any of the material contained herein, may be reproduced without written consent of the author.

#### **Disclaimer**

The information in this document is subject to change without notice. The manufacturer makes no representations or warranties with respect to the contents hereof and specifically disclaim any implied warranties of merchantability or fitness for any particular purpose. The manufacturer reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of the manufacturer to notify any person of such revision or changes.

#### **Limitations of Liability**

UNDER NO CIRCUMSTANCES SHALL CLAVISTER OR ITS SUPPLIERS BE LIABLE FOR DAMAGES OF ANY CHARACTER (E.G. DAMAGES FOR LOSS OF PROFIT, SOFTWARE RESTORATION, WORK STOPPAGE, LOSS OF SAVED DATA OR ANY OTHER COMMERCIAL DAMAGES OR LOSSES) RESULTING FROM THE APPLICATION OR IMPROPER USE OF THE CLAVISTER PRODUCT OR FAILURE OF THE PRODUCT, EVEN IF CLAVISTER IS INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. FURTHERMORE, CLAVISTER WILL NOT BE LIABLE FOR THIRD-PARTY CLAIMS AGAINST CUSTOMER FOR LOSSES OR DAMAGES. CLAVISTER WILL IN NO EVENT BE LIABLE FOR ANY DAMAGES IN EXCESS OF THE AMOUNT CLAVISTER RECEIVED FROM THE END-USER FOR THE PRODUCT.

---

---

---

---

## Table of Contents

Preface .....	xii
1. Product Overview .....	1
1.1. About Clavister CorePlus .....	1
1.2. CorePlus Architecture .....	3
1.2.1. State-based Architecture .....	3
1.2.2. CorePlus Building Blocks .....	3
1.2.3. Basic Packet Flow .....	3
1.3. CorePlus Packet Flow .....	6
2. Operations and Maintenance .....	10
2.1. Configuring CorePlus .....	10
2.1.1. Overview .....	10
2.1.2. User Accounts .....	10
2.1.3. Command Line Interface (CLI) .....	11
2.1.4. Working with Configurations .....	12
2.1.5. NetCon .....	15
2.2. Events and Logging .....	16
2.2.1. Overview .....	16
2.2.2. Event Messages .....	16
2.2.3. Event Message Distribution .....	16
2.2.4. Customizing Event Message Distribution .....	19
2.3. RADIUS Accounting .....	21
2.3.1. Overview .....	21
2.3.2. RADIUS Accounting messages .....	21
2.3.3. Interim Accounting Messages .....	23
2.3.4. Activating RADIUS Accounting .....	23
2.3.5. RADIUS Accounting Security .....	23
2.3.6. RADIUS Accounting and High Availability .....	23
2.3.7. Handling Unresponsive Servers .....	24
2.3.8. Accounting and System Shutdowns .....	24
2.3.9. Limitations with NAT'ed Networks .....	24
2.4. Monitoring .....	25
2.4.1. Real-time Monitoring Counters .....	25
2.5. Maintenance .....	28
2.5.1. Firmware Upgrades .....	28
2.5.2. Reset to Factory Defaults .....	29
2.5.3. Configuration Backup and Restore .....	29
2.5.4. Auto-Update Mechanism .....	29
3. Fundamentals .....	32
3.1. The Address Book .....	32
3.1.1. Overview .....	32
3.1.2. IP Addresses .....	32
3.1.3. Ethernet Addresses .....	34
3.1.4. Address Groups .....	34
3.1.5. Auto-Generated Address Objects .....	34
3.2. Services .....	36
3.2.1. Overview .....	36
3.2.2. TCP and UDP Based Services .....	36
3.2.3. ICMP Services .....	38
3.2.4. Custom IP Protocol Services .....	39
3.3. Interfaces .....	40
3.3.1. Overview .....	40
3.3.2. Ethernet .....	41
3.3.3. Virtual LAN .....	43
3.3.4. PPPoE .....	44
3.3.5. GRE Tunnels .....	46
3.3.6. Loopback Interfaces .....	46
3.3.7. Interface Groups .....	47

---

3.4. ARP .....	48
3.4.1. Overview .....	48
3.4.2. ARP in CorePlus .....	48
3.4.3. ARP Cache .....	48
3.4.4. Static and Published ARP Entries .....	49
3.4.5. Advanced ARP Settings .....	51
3.5. The IP Rule-Set .....	53
3.5.1. Overview .....	53
3.5.2. Rule Evaluation .....	53
3.5.3. Rule components .....	54
3.6. Schedules .....	56
3.7. X.509 Certificates .....	58
3.7.1. Overview .....	58
3.7.2. The Certification Authority .....	58
3.7.3. Validity Time .....	58
3.7.4. Certificate Revocation Lists .....	58
3.7.5. Trusting Certificates .....	59
3.7.6. Identification Lists .....	59
3.7.7. X.509 Certificates in CorePlus .....	59
3.8. System Settings .....	60
3.8.1. Date and Time .....	60
3.8.2. DNS .....	63
4. Routing .....	65
4.1. Overview .....	65
4.2. Static Routing .....	66
4.2.1. Static Routing in CorePlus .....	67
4.2.2. Route Failover .....	70
4.2.3. Proxy ARP .....	72
4.3. Policy-based Routing .....	73
4.3.1. Overview .....	73
4.3.2. Policy-based Routing Tables .....	73
4.3.3. Policy-based Routing Rules .....	74
4.3.4. Interface/Routing Table Association .....	74
4.3.5. Policy-based Routing Table Selection .....	74
4.3.6. The <i>Ordering</i> parameter .....	74
4.4. Virtual Routing .....	78
4.4.1. Overview .....	78
4.4.2. The Problem .....	78
4.4.3. Benefits of Virtual Routing .....	79
4.4.4. Policies in Virtual Systems .....	81
4.4.5. Trouble Shooting .....	82
4.5. Dynamic Routing .....	84
4.5.1. Dynamic Routing overview .....	84
4.5.2. OSPF .....	85
4.5.3. Dynamic Routing Policy .....	88
4.6. Multicast Routing .....	91
4.6.1. Overview .....	91
4.6.2. Multicast Forwarding using the SAT Multiplex Rule .....	91
4.6.3. IGMP Configuration .....	94
4.7. Transparent Mode .....	99
4.7.1. Overview of Transparent Mode .....	99
4.7.2. Comparison with Routing mode .....	99
4.7.3. Transparent Mode implementation .....	99
4.7.4. Enabling Transparent Mode .....	100
4.7.5. Transparent Mode example scenarios .....	100
4.7.6. Spanning Tree BPDU Support .....	104
5. DHCP Services .....	107
5.1. Overview .....	107
5.2. DHCP Servers .....	108
5.3. DHCP Relaying .....	109
6. Security Mechanisms .....	111
6.1. Access Rules .....	111
6.1.1. Introduction .....	111

---

---

6.1.2. IP spoofing .....	111
6.1.3. Access Rule Settings .....	112
6.2. Application Layer Gateways .....	114
6.2.1. Overview .....	114
6.2.2. Hyper Text Transfer Protocol .....	114
6.2.3. File Transfer Protocol .....	114
6.2.4. Simple Mail Transfer Protocol .....	119
6.2.5. H.323 .....	120
6.3. Intrusion Detection and Prevention .....	134
6.3.1. Overview .....	134
6.3.2. IDP Rules .....	134
6.3.3. IDP Pattern Matching .....	135
6.3.4. IDP Signature Groups .....	136
6.3.5. IDP Actions .....	137
6.3.6. Subscribing to Clavister IDP .....	137
6.4. Anti-Virus .....	140
6.4.1. Overview .....	140
6.4.2. Implementation .....	140
6.4.3. Activation .....	141
6.4.4. The Signature Database .....	141
6.4.5. Subscribing to the Clavister Anti-Virus Service .....	141
6.4.6. Anti-Virus Options .....	142
6.5. Web Content Filtering .....	144
6.5.1. Overview .....	144
6.5.2. Active Content Handling .....	144
6.5.3. Static Content Filtering .....	145
6.5.4. Dynamic Content Filtering .....	146
6.6. Denial-Of-Service Attacks (DoS) .....	158
6.6.1. About Denial of Service attacks .....	158
6.6.2. The Ping of Death and Jolt attacks .....	159
6.6.3. Fragmentation overlap attacks: Teardrop, Bonk, Boink and Nestea .....	159
6.6.4. The Land and LaTierra attacks .....	159
6.6.5. The WinNuke attack .....	159
6.6.6. Amplification attacks: Smurf, Papasmurf, Fraggle .....	160
6.6.7. The TCP SYN Flood attack .....	161
6.6.8. The Jolt2 attack .....	161
6.6.9. Distributed Denial of Service attacks .....	161
6.7. Blacklisting Hosts and Networks .....	163
7. Address Translation .....	165
7.1. Dynamic Address Translation (NAT) .....	165
7.1.1. Which Protocols can NAT handle? .....	166
7.2. Static Address Translation (SAT) .....	168
7.2.1. Translation of a Single IP Address (1:1) .....	168
7.2.2. Translation of Multiple IP Addresses (M:N) .....	171
7.2.3. All-to-One Mappings (N:1) .....	173
7.2.4. Port Translation .....	173
7.2.5. Which Protocols can SAT handle? .....	174
7.2.6. Which SAT Rule is executed if several are matching? .....	174
7.2.7. SAT and FwdFast Rules .....	175
8. User Authentication .....	178
8.1. Overview .....	178
8.2. Authentication Components .....	180
8.3. Authentication Process .....	182
9. Virtual Private Networks .....	184
9.1. VPN overview .....	184
9.1.1. The need for VPNs .....	184
9.1.2. The basics of VPN Encryption .....	184
9.1.3. Planning a VPN .....	184
9.2. IPsec .....	186
9.2.1. IPsec Basics .....	186
9.2.2. Proposal Lists .....	195
9.2.3. Pre-shared Keys .....	196
9.2.4. Identification Lists .....	197

---

---

9.3. IPsec tunnels .....	199
9.3.1. Overview of IPsec tunnels .....	199
9.3.2. LAN to LAN tunnels with a Pre-shared Key .....	199
9.3.3. Roaming Clients .....	200
9.3.4. Fetching CRLs from an alternate LDAP server .....	205
9.3.5. Troubleshooting with <i>ikesnoop</i> .....	206
9.4. PPTP/L2TP .....	214
9.4.1. PPTP .....	214
9.4.2. L2TP .....	215
10. Traffic Management .....	220
10.1. Traffic Shaping .....	220
10.1.1. Introduction .....	220
10.1.2. Traffic Shaping Basics .....	220
10.1.3. Traffic Shaping in CorePlus .....	221
10.1.4. Pipes Basics .....	222
10.1.5. Priorities and Guarantees .....	225
10.1.6. Grouping Users of a Pipe .....	229
10.2. Traffic Rate Limiting with Threshold Rules .....	232
10.2.1. Rate limit blacklisting .....	232
10.3. Server Load Balancing .....	233
10.3.1. Introduction .....	233
10.3.2. Server Load Balancing Basics .....	234
11. High Availability .....	239
11.1. Overview .....	239
11.2. How rapid failover is accomplished .....	241
11.2.1. Shared IP addresses and Failover .....	241
11.2.2. Cluster heartbeats .....	241
11.2.3. The synchronization interface .....	242
11.3. Things to keep in mind .....	243
11.3.1. Communicating with two gateways .....	243
11.3.2. Configuration issues .....	243
12. Advanced Settings .....	245
12.1. IP Level Settings .....	246
12.2. TCP Level Settings .....	249
12.3. ICMP Level Settings .....	253
12.4. ARP Settings .....	254
12.5. Stateful Inspection Settings .....	257
12.6. Connection Timeouts .....	259
12.7. Size Limits by Protocol .....	260
12.8. Fragmentation Settings .....	262
12.9. Local Fragment Reassembly Settings .....	266
12.10. VLAN Settings .....	267
12.11. SNMP Settings .....	268
12.12. DHCP Settings .....	269
12.13. DHCPRelay Settings .....	270
12.14. DHCP Server Settings .....	271
12.15. IPsec Settings .....	272
12.16. Transparent Mode Settings .....	273
12.17. Logging Settings .....	275
12.18. High Availability Settings .....	276
12.19. Time Synchronization Settings .....	277
12.20. DNS Client Settings .....	279
12.21. Remote Administration Settings .....	280
12.22. HTTP Poster Settings .....	281
12.23. Hardware Performance Settings .....	282
12.24. PPP Settings .....	283
12.25. RADIUS Accounting Settings .....	284
12.26. IDP .....	285
12.27. Multicast Settings .....	286
12.28. Hardware Monitor Settings .....	288
12.29. Packet Re-assembly Settings .....	290
12.30. Miscellaneous Settings .....	291
A. Subscribing to Security Updates .....	293

---

B. CLI Reference .....	296
C. <b>fwctl</b> command options .....	314
D. IDP Signature Groups .....	316
E. Anti-Virus MIME filetypes .....	320
F. The OSI framework .....	324
Alphabetical Index .....	325



---

## List of Figures

1.1. Packet Flow Schematic Part I .....	6
1.2. Packet Flow Schematic Part II .....	7
1.3. Packet Flow Schematic Part III .....	8
2.1. CorePlus Firmware Structure .....	28
4.1. A Route Failover Scenario for ISP Access .....	70
4.2. Virtual Links Example 1 .....	87
4.3. Virtual Links Example 2 .....	88
4.4. Multicast Forwarding - No Address Translation .....	92
4.5. Multicast Forwarding - Address Translation .....	93
4.6. Multicast Snoop .....	95
4.7. Multicast Proxy .....	95
4.8. Transparent mode scenario 1 .....	100
4.9. Transparent mode scenario 2 .....	102
4.10. An example BPDU relaying scenario .....	104
6.1. IDP Database Updating .....	138
6.2. Dynamic Content Filtering Flow .....	147
9.1. The AH protocol .....	193
9.2. The ESP protocol .....	193
10.1. Packet flow through pipes .....	222
10.2. The Eight Pipe Precedences. ....	225
10.3. A Pipe defined with minimum precedence and maximum precedence. ....	226
10.4. Example of a pipe with traffic in one precedence, grouped per IP address. ....	230
10.5. A Server Load Balancing configuration .....	233
10.6. Connections from Three Clients .....	235
10.7. Stickiness and Round-Robin .....	236
10.8. Stickiness and Connection Rate .....	236
11.1. High Availability Setup Example .....	240
F.1. The 7 layers of the OSI model .....	324

---

## List of Examples

1. Example notation .....	xii
2.1. Listing Configuration Objects .....	13
2.2. Displaying a Configuration Object .....	13
2.3. Editing a Configuration Object .....	13
2.4. Adding a Configuration Object .....	14
2.5. Deleting a Configuration Object .....	14
2.6. Activating and Committing a Configuration .....	15
2.7. Enable Logging to a Syslog Host .....	17
2.8. Enabling Logging to Clavister Loggers .....	18
2.9. Sending SNMP Traps to an SNMP Trap Receiver .....	19
2.10. Upgrading the CorePlus Core .....	28
3.1. Adding an IP Host .....	33
3.2. Adding an IP Network .....	33
3.3. Adding an IP Range .....	33
3.4. Deleting an Address Object .....	34
3.5. Listing the Available Services .....	36
3.6. Viewing a Specific Service .....	36
3.7. Adding a TCP/UDP Service .....	37
3.8. Adding a IP Protocol Service .....	39
3.9. Enabling DHCP .....	43
3.10. Defining a virtual LAN .....	44
3.11. Configuring a PPPoE client on the wan interface with traffic routed over PPPoE. ....	45
3.12. Creating a Loopback Interface Pair .....	46
3.13. Creating an Interface Group .....	47
3.14. Displaying the ARP Cache .....	49
3.15. Flushing the ARP Cache .....	49
3.16. Defining a Static ARP Entry .....	50
3.17. Setting up a Time-Scheduled Policy .....	56
3.18. Setting the Current Date and Time .....	60
3.19. Setting the Time Zone .....	60
3.20. Enabling DST .....	61
3.21. Enabling Time Synchronization using SNTP .....	61
3.22. Manually Triggering a Time Synchronization .....	62
3.23. Modifying the Maximum Adjustment Value .....	62
3.24. Forcing Time Synchronization .....	62
3.25. Configuring DNS Servers .....	63
4.1. Displaying the Routing Table .....	68
4.2. Displaying the Core Routes .....	69
4.3. Creating a Policy-Based Routing table .....	75
4.4. Creating the Route .....	75
4.5. Policy Based Routing Configuration .....	76
4.6. Importing Routes from an OSPF AS into the Main Routing Table .....	89
4.7. Exporting the Default Route into an OSPF AS .....	90
4.8. Forwarding of Multicast Traffic using the SAT Multiplex Rule .....	93
4.9. Multicast Forwarding - Address Translation .....	94
4.10. IGMP - No Address Translation .....	96
4.11. Configuration if1 .....	97
4.12. Configuration if2 - Group Translation .....	98
4.13. Setting up Transparent Mode - Scenario 1 .....	101
4.14. Setting up Transparent Mode - Scenario 2 .....	103
5.1. Setting up a DHCP server .....	108
5.2. Setting up a DHCP relay .....	109
6.1. Setting up an Access Rule .....	112
6.2. Protecting an FTP Server with ALG .....	115
6.3. Protecting FTP Clients .....	118
6.4. Protecting Phones Behind Clavister Security Gateways .....	122
6.5. H.323 with private IP addresses .....	123
6.6. Two Phones Behind Different Clavister Security Gateways .....	124

---

6.7. Using Private IP Addresses .....	125
6.8. H.323 with Gatekeeper .....	127
6.9. H.323 with Gatekeeper and two Clavister Security Gateways .....	128
6.10. Using the H.323 ALG in a Corporate Environment .....	129
6.11. Configuring remote offices for H.323 .....	132
6.12. Allowing the H.323 Gateway to register with the Gatekeeper .....	132
6.13. Setting up IDP for a Mail Server .....	138
6.14. Enabling Anti-Virus Scanning .....	142
6.15. Stripping ActiveX and Java applets .....	145
6.16. Setting up a white and blacklist .....	146
6.17. Enable Dynamic Content Filtering .....	148
6.18. Enabling Audit Mode .....	149
6.19. Reclassifying a blocked site .....	150
6.20. Adding a whitelist entry .....	163
7.1. Adding a NAT Policy .....	166
7.2. Enabling Traffic to a Protected Web Server in a DMZ .....	168
7.3. Enabling Traffic to a Web Server on an Internal Network .....	170
7.4. Translating Traffic to Multiple Protected Web Servers .....	171
8.1. Creating an authentication user group .....	183
9.1. Using a Proposal List .....	195
9.2. Using a Pre-Shared key .....	196
9.3. Using an Identity List .....	197
9.4. Setup a LAN to LAN VPN tunnel with PSK .....	200
9.5. Setting up a PSK based VPN tunnel for roaming clients .....	201
9.6. Setting up a Self-signed Certificate based VPN tunnel for roaming clients .....	202
9.7. Setting up a CA Server issued Certificate based VPN tunnel for roaming clients .....	203
9.8. Setting up an LDAP server .....	205
9.9. Setting up a PPTP server .....	214
9.10. Setting up an L2TP server .....	215
9.11. Setting up an L2TP Tunnel .....	215
10.1. Applying a Simple Bandwidth Limit .....	222
10.2. Applying a Two-Way Bandwidth Limit .....	224

---

# Preface

## Intended audience

The target audience for this reference guide is Administrators who are responsible for configuring and managing Clavister Security Gateways which are running the CorePlus operating system. This guide assumes that the reader has some basic knowledge of networks and network security.

## Text structure and conventions

The text is broken down into chapters and subsections. Numbered subsections are shown in the table of contents at the beginning. An index is included at the end of the document to aid with alphabetical lookup of subjects.

Where a "See section" link (such as: see Chapter 9, *Virtual Private Networks* ) is provided in the main text this can be clicked to take the reader directly to that reference.

Text that may appear in the user interface of the product is designated by being in **Bold Case**. Where a term is being introduced for the first time or being stressed *it may appear in italics*.

Where console interaction is shown in the main text outside of an example this will appear in a box with a gray background:

```
Cmd>
```

Where a web address reference is shown in the text this will open the specified URL in a browser in a new window when clicked (some systems may not allow this). For example: <http://www.clavister.com>.

## Examples

Examples in the text are denoted by the header **Example** and appear with a gray background as shown below. They contain a Clavister FineTune example and/or a CLI example.

### Example 1. Example notation

Information about what the example is trying to achieve is found here, sometimes with an explanatory image.

#### **Clavister FineTune**

The Clavister FineTune example appears here and is described as a series of steps:

1. First step
2. Second step

The term *context menu* refers to the menu that appears by right-clicking a menu item.

## Notes to the main text

Special sections of text which the reader should pay special attention to are indicated by icons on the left hand side of the page followed by a short paragraph in italicized text. Such sections have the following types and purposes:



### **Note**

*This indicates some piece of information that is an addition to the preceding text. It*

*may concern something that is being emphasised or something that is not obvious or explicitly stated in the preceding text.*

**Tip**

*This indicates a piece of non-critical information that is useful to know in certain situations but is not essential reading.*

**Caution**

*This indicates where the reader should be careful with their actions as an undesirable situation may result if care is not exercised.*

**Important**

*This is an essential point that the reader should read and understand.*

**Warning**

*This is essential reading for the user as they should be aware that a serious situation may result if certain actions are taken or not taken.*

---

# Chapter 1. Product Overview

This chapter outlines the key features of CorePlus.

- About Clavister CorePlus, page 1
- CorePlus Architecture, page 3
- CorePlus Packet Flow, page 6

## 1.1. About Clavister CorePlus

Clavister CorePlus is the firmware, the software engine that drives and controls all Clavister Security Gateway products. CorePlus can also run on your preferred choice of server hardware.

Designed as a network security operating system, CorePlus features high throughput performance with high reliability plus super-granular control. In contrast to products built on standard operating systems such as Unix or Microsoft Windows, CorePlus offers seamless integration of all subsystems, in-depth administrative control of all functionality as well as a minimal attack surface which helps negate the risk of being a target for security attacks.

From the administrator's perspective the conceptual approach of CorePlus is to visualize operations through a set of logical building blocks or objects, which allow the configuration of the product in an almost limitless number of different ways. This granular control allows the administrator to meet the requirements of the most demanding network security scenario.

CorePlus is an extensive and feature-rich network operating system. The list below presents the most essential features:

### **IP Routing**

CorePlus provides a variety of options for IP routing including static routing, dynamic routing, multicast routing and advanced virtual routing capabilities. In addition, CorePlus supports features such as Virtual LANs, Route Monitoring, Proxy ARP and Transparency. For more information, please see Chapter 4, *Routing*.

### **Address Translation**

For functionality as well as security reasons, CorePlus supports policy-based address translation. Dynamic Address Translation (NAT) as well as Static Address Translation (SAT) is supported, and resolves most types of address translation needs. This feature is covered in Chapter 7, *Address Translation*.

### **Firewalling**

At the heart of the product, CorePlus features stateful inspection-based firewalling for common protocols such as TCP, UDP and ICMP. As an administrator, you have the possibility to define detailed firewalling policies based on source and destination network and interface, protocol, ports, user credentials, time-of-day and much more. Section 3.5, "The IP Rule-Set", describes how to use the firewalling aspects of CorePlus.

### **Intrusion Detection and Prevention**

To mitigate application-layer attacks towards vulnerabilities in services and applications, CorePlus provides a powerful Intrusion Detection and Prevention (IDP) engine. The IDP engine is policy-based and is able to perform high-performance scanning and detection of attacks and can perform blocking and optional black-listing of attacking hosts. For more in-

---

	formation about the IDP capabilities of CorePlus, please see Section 6.3, “Intrusion Detection and Prevention”.
<b>Web Content Filtering</b>	CorePlus provides various mechanisms for filtering web content that is deemed inappropriate according to your web usage policy. Web content can be blocked based on category, malicious objects can be removed and web sites can be whitelisted or blacklisted in multiple policies. For more information, please see Section 6.5, “Web Content Filtering”.
<b>Virtual Private Networking</b>	A device running CorePlus is highly suitable for participating in a Virtual Private Network (VPN). CorePlus supports IPsec, L2TP and PPTP based VPNs concurrently, can act as either server or client for all of the VPN types, and can provide individual security policies for each VPN tunnel. Virtual Private Networking is covered in detail by Chapter 9, <i>Virtual Private Networks</i> .
<b>Traffic Management</b>	With the support of Traffic Shaping, Threshold Rules and Server Load Balancing features, CorePlus is optimal for traffic management. The Traffic Shaping feature enables fine-granular limiting and balancing of bandwidth; Threshold Rules allows for implementing various types of thresholds where to alarm or limit network traffic, and Server Load Balancing enables a device running CorePlus to distribute network load to multiple hosts. Chapter 10, <i>Traffic Management</i> , provides more detailed information on the various traffic management capabilities.
<b>User Authentication</b>	A CorePlus device can be used for authenticating users before allowing access to protected resources. Multiple local user databases are supported as well as multiple external RADIUS servers, and separate authentication policies can be defined to support separate authentication schemes for different kinds of traffic. Chapter 8, <i>User Authentication</i> , provides detailed information about this feature.
<b>Operations and Maintenance</b>	To facilitate management of a CorePlus device, administrative control is enabled through Clavister FineTune or via the Command Line Interface. In addition, CorePlus provides very detailed event and logging capabilities and support for monitoring using standards such as SNMP. For more information, please see Chapter 2, <i>Operations and Maintenance</i> .
<b>High Availability</b>	High Availability is supported through automatic fault-tolerant fail-over to a secondary CorePlus device. This feature is described in more detail in Chapter 11, <i>High Availability</i> .

In addition to the above mentioned features, CorePlus includes a number of features such as accounting using RADIUS Accounting, providing DHCP services, protection against Denial-of-Service (DoS) attacks, support for PPPoE, GRE, dynamic DNS services and much more.

Reading through this documentation carefully will ensure that you get the most out of your CorePlus product. In addition to this document, the reader should also be aware of the companion volumes:

- The Clavister FineTune Administration Guide which describes how to use the Clavister FineTune application.
- The CorePlus Log Reference Guide which details all CorePlus log event messages.

These documents together form the essential documentation for CorePlus operation.

## 1.2. CorePlus Architecture

### 1.2.1. State-based Architecture

The CorePlus architecture is centered around the concept of state-based connections. Traditional IP routers or switches commonly inspect all packets and then perform forwarding decisions based on information found in the packet headers. With this approach, packets are forwarded without any sense of context which basically eliminates any possibility to detect and analyze complex protocols and enforce corresponding security policies.

A CorePlus device, on the contrary, will inspect and forward traffic on a per-connection basis. In other words, CorePlus is able to detect when a new connection is being established, and then keeps a small piece of information, a "state", for the entire life-length of that connection. By doing this, CorePlus is able to understand the context of the network traffic, which enables the device to perform in-depth traffic scanning, apply bandwidth management and much more. In addition, this approach provides high throughput performance with the added advantage of a design that is highly scalable.

### 1.2.2. CorePlus Building Blocks

The basic building blocks in CorePlus are interfaces, logical objects and various types of rules (or rule-sets).

*Interfaces* are the doorways for network traffic passing through, to or from the system. Without interfaces, a CorePlus system has no means for receiving or sending traffic. Several types of interfaces are supported; Physical Interfaces, Physical Sub-Interfaces and Tunnel Interfaces. *Physical interfaces* corresponds to actual physical Ethernet ports; *physical sub-interfaces* include VLAN and PPPoE interfaces while *tunnel interfaces* are used for receiving and sending traffic in VPN tunnels.

The CorePlus interface design is symmetric, meaning that the interfaces of the device are not fixed as being on the "insecure outside" or "secure inside" of a network topology. The notion of what is inside and outside is totally for the administrator to define.

*Logical objects* can be seen as pre-defined building blocks for use by the rule-sets. The address book, for instance, contains named objects representing host and network addresses. Another example of logical objects are services, representing specific protocol and port combinations. Also important objects are the Application Layer Gateway (ALG) objects, used for defining additional parameters on specific protocols such as HTTP, FTP and H.323.

Finally, the various *rule-sets* are used for actually implementing the policies in the system. The most fundamental rule-set is the IP Rules, which is used to define the layer 3 IP filtering policy as well as carrying out address translation and server load balancing. The Traffic Shaping Rules define the policy for bandwidth management, the IPS Rules controls the behavior of the intrusion prevention engine and so forth.

### 1.2.3. Basic Packet Flow

This section outlines the basic flow for packets received and forwarded by a CorePlus device. Please note that this description is simplified to ease the understanding and might not be fully applicable in all scenarios. The basic principle, however, is still valid in all applications.

1. An Ethernet frame is received on one of the Ethernet interfaces in the system. Basic Ethernet frame validation is performed and the packet is dropped if the frame is invalid.
2. The packet is associated with a Source Interface. The source interface is determined as follows:
  - If the Ethernet frame contains a VLAN ID (Virtual LAN identifier), the system checks for a configured VLAN interface with a corresponding VLAN ID. If one is found, that VLAN interface becomes the source interface for the packet. If no matching interface is found, the



packet is dropped and the event is logged.

- If the Ethernet frame contains a PPP payload, the system checks for a matching PPPoE interface. If one is found, that interface becomes the source interface for the packet. If no matching interface is found, the packet is dropped and the event is logged.
  - If none the above is true, the receiving Ethernet interface becomes the source interface for the packet.
3. The IP datagram within the packet is passed on to the CorePlus Consistency Checker. The consistency checker performs a number of sanity checks on the packet, including validation of checksums, protocol flags, packet length and so forth. If the consistency checks fail, the packet gets dropped and the event is logged.
  4. CorePlus now tries to lookup an existing connection by matching parameters from the incoming packet. A number of parameters are used in the match attempt, including the source interface, source and destination IP addresses, IP protocol and so forth.

If a match cannot be found, a connection establishment process starts which includes steps 5 to 10 below. If a match is found, the forwarding process continues at step 11 below.

5. The source interface is examined to find out if the interface is a member of a specific routing table. Also, the Policy-based Routing Rules are evaluated to determine the correct routing table for the connection.
6. The Access rules are evaluated to find out if the source IP address of the new connection is allowed on the received interface. If no access rule matches then a reverse route lookup will be done. In other words, by default, an interface will only accept source IP addresses that belong to networks routed over that interface. If the access rules or the reverse route lookup determine that the source IP is invalid, then the packet is dropped and the event is logged.
7. A route lookup is being made using the appropriate routing table. The destination interface for the connection has now been determined.
8. The IP rules are now searched for a rule that matches the packet. Basically, the following parameters are part of the matching process: Source and destination interfaces, source and destination network, IP protocol (TCP, UDP, ICMP etc), TCP/UDP ports or ICMP types and schedule (time-of-day).

If a match cannot be found, the packet is dropped.

If a rule is found that matches the new connection, the Action parameter of the rule decides what CorePlus should do with the connection. If the action is Drop, the packet is dropped and the event is logged according to the log settings for the rule.

If the action is Allow, the packet is allowed through the system. A corresponding state will be added to the connection table for matching subsequent packets belonging to the same connection. In addition, the Service object which matched the IP protocol and ports might have contained a reference to an Application Layer Gateway (ALG) object. This information is recorded in the state so that CorePlus will know that application layer processing will have to be performed on the connection.

Finally, the opening of the new connection will be logged according to the log settings of the rule.



**Note**

*There are actually a number of additional actions available such as address translation and server load balancing. The basic concept of dropping and allowing traffic is still the same.*

9. The Intrusion Detection and Prevention (IDP) Rules are now evaluated in a similar way to the

IP rules. If a match is found, the IDP data is recorded with the state. By doing this, CorePlus will know that IDP scanning is supposed to be conducted on all packets belonging to this connection.

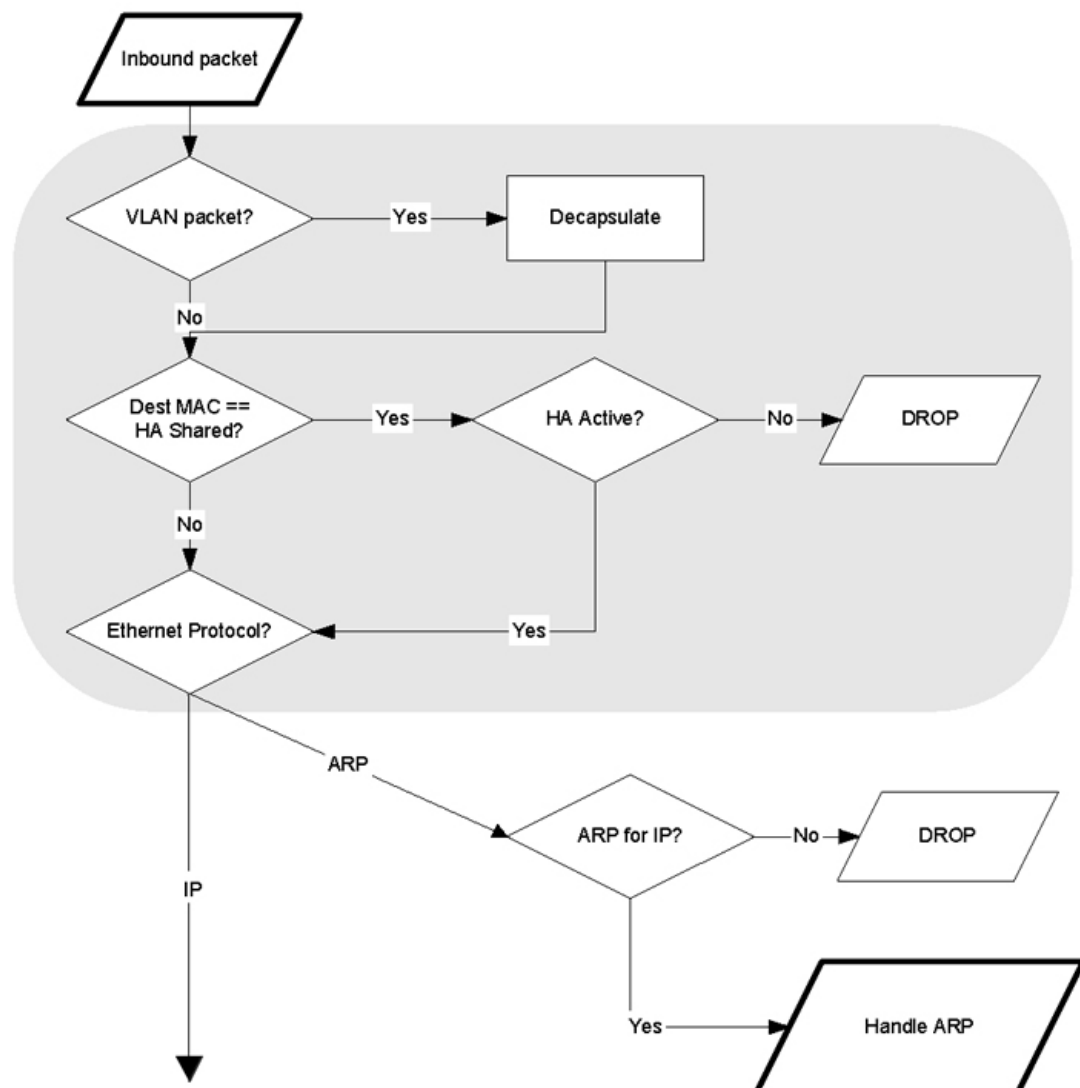
10. The Traffic Shaping and the Rate Limiting Rule-sets are now searched. If a match is found, the corresponding information is recorded with the state. This will enable proper traffic management on the connection.
11. From the information in the state, CorePlus now knows what to do with the incoming packet:
  - If ALG information is present or if IDP scanning is to be performed, the payload of the packet is taken care of by the TCP Pseudo-Reassembly subsystem, which in turn makes use of the different Application Layer Gateways, layer 7 scanning engines and so forth, to further analyze or transform the traffic.
  - If the contents of the packet is encapsulated (i.e. being IPsec, L2TP/PPTP or some other type of tunneled traffic), the interface lists are checked for a matching interface. If one is found, the packet is decapsulated and the payload (the plaintext) is sent into CorePlus again, now with source interface being the matched tunnel interface. In other words, the process continues at step 3 above.
  - If traffic management information is present, the packet might get queued or otherwise be subjected to actions related to traffic management.
12. Eventually, the packet will be forwarded out on the destination interface according to the state. If the destination interface is a tunnel interface or a physical sub-interface, additional processing such as encryption, and encapsulation and so forth might occur.

The following section provides a set of diagrams which illustrate the flow of packets through CorePlus.

## 1.3. CorePlus Packet Flow

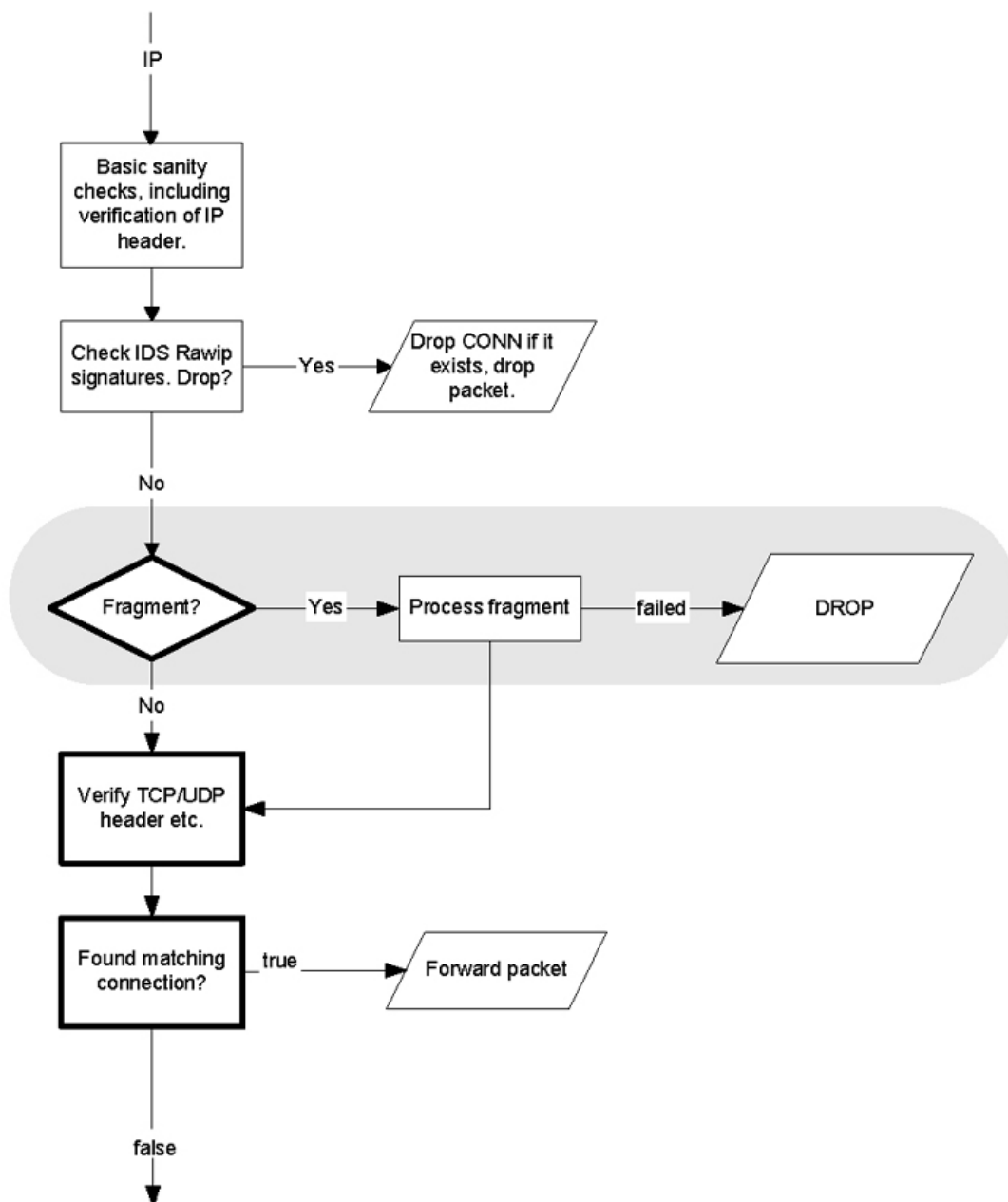
The diagrams in this section provide a summary of the flow of packets through the CorePlus state-engine. There are three diagrams, each flowing into the next.

**Figure 1.1. Packet Flow Schematic Part I**



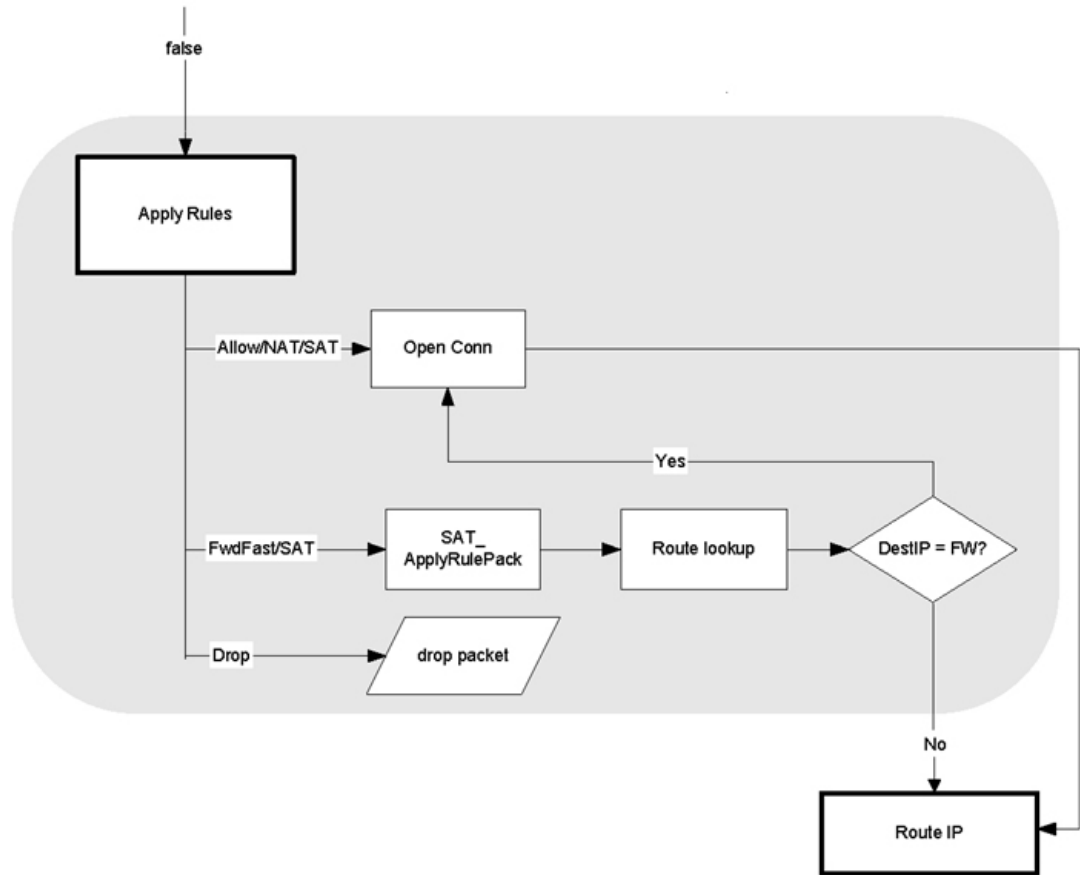
The packet flow is continued on the following page.

Figure 1.2. Packet Flow Schematic Part II



The packet flow is continued on the following page.

Figure 1.3. Packet Flow Schematic Part III





---

# Chapter 2. Operations and Maintenance

This chapter describes the operations and maintenance related aspects of CorePlus. In this chapter, the various management interfaces will be presented, the concept of events and logging will be described, and functionality such as accounting and monitoring will be introduced. Also, pure maintenance tasks, such as firmware upgrades, backup and restore of configurations and so forth will be covered.

- Configuring CorePlus, page 10
- Events and Logging, page 16
- RADIUS Accounting, page 21
- Monitoring, page 25
- Maintenance, page 28

## 2.1. Configuring CorePlus

### 2.1.1. Overview

CorePlus is designed to be give both high performance and high reliability. Not only does it provide an extensive feature set, it also enables the administrator to be in full control of almost every detail of the system. This means the product can be deployed in the most challenging environments.

A good understanding on how CorePlus configuration is performed is crucial for proper usage of the system. For this reason, this section provides an in-depth presentation of how the configuration subsystem is designed as well as a detailed description on how to work with the various types of management interfaces that are provided by the product.

CorePlus provides the following management interfaces:

<b>Clavister FineTune</b>	Clavister FineTune is Microsoft Windows based software used for centralized management of several Clavister Security Gateways. Please refer to the separate Clavister FineTune user guide for a more detailed description of this tool.
<b>Command Line Interface (CLI)</b>	The Command Line Interface, accessible locally via serial console port or remotely using the NetCon protocol, provides a text-based console where a number of tasks can be carried out. Please note that the current CorePlus version does not support configuration using the CLI.

Access to a management interface is regulated by a *remote management policy*, where you can restrict management access based on source network, source interface, credentials and so forth.

During the *Setup Base Configuration* phase, you select a management interface and provide IP address information for that interface. As a result of this, a remote management policy is added allowing NetCon access from hosts on the specified management network.

### 2.1.2. User Accounts

CorePlus offers several possibilities for storing user information, either using local user databases or external databases. There are no default user accounts predefined and these must be set up. For more detailed information about user authentication, please see Chapter 8, *User Authentication*.

## 2.1.3. Command Line Interface (CLI)

The CLI provides a number of commands used for monitoring and troubleshooting. The CLI is available either locally through the serial console port, or remotely using the NetCon protocol.

**Note**

*The current implementation of CorePlus does not support configuration through the CLI.*

The CLI provides a comprehensive set of commands that allow the displaying of configuration data, showing runtime data from the system as well as performing system maintenance tasks.

For a complete reference to all CLI commands, please see the Clavister *CLI Reference Guide*.

### 2.1.3.1. CLI Access Methods

#### Serial Console Port

The serial console port is an RS-232 port that enables access to the CLI through a serial connection to a PC or terminal. To locate the serial console port on your Clavister system, please see the corresponding hardware installation guide.

To use the console port, you need the following equipment:

- A terminal or a (portable) computer with a serial port and the ability to emulate a terminal (i.e. using the Hyper Terminal software included in most Microsoft Windows installations). The serial console port uses the following default settings: *9600 baud, No parity, 8 bits and 1 stop bit*.
- An RS-232 cable with appropriate connectors. An appliance package includes an RS-232 null-modem cable.

To connect a terminal to the console port, follow these steps:

1. Set the terminal protocol as described previously.
2. Connect one of the connectors of the RS-232 cable directly to the console port on your system hardware.
3. Connect the other end of the cable to the terminal or the serial connector of the computer running the communications software.
4. Press the *enter* key on the terminal. The CorePlus login prompt should appear on the terminal screen.

#### NetCon

The NetCon protocol is the Clavister proprietary protocol that can be used to access the CLI. A NetCon connection has the advantage of being encrypted and offers a secure means of remote access.

There are two primary means of accessing CorePlus via the NetCon protocol:

**The Remote Console** This is a console which is part of Clavister FineTune. It has full access to the Clavister FineTune configuration database.

**fwctl** A stand-alone console command line tool. This is available in two versions, one for Linux and one for Microsoft Windows. **fwctl** was primarily designed for Linux environments and so follows Linux command line conventions (Appendix C, *fwctl command options* contains



a full list of commands). It has the limitation that it does not have direct access to the Clavister FineTune configuration database, however it is able to upload complete configurations to the gateway as XML configuration files.

There is no specific remote management policy needed for CLI access using NetCon. As long as you are allowed to connect to the gateway using NetCon, CLI access will be granted as well.

**Note**

Where the hardware supports attachment of a VGA display and a keyboard, the CLI can also be accessed using such a console.

## 2.1.3.2. Common CLI Operations

### Logging on to the CLI

When access to the CLI has been established using one of the methods described in the earlier sections, you need to logon to the system before being able to execute any CLI command. This authentication step is needed to ensure that only trusted users can access the system, as well as providing user information for the audit mechanism.

The CLI uses the common user authentication mechanisms provided. In other words, local user databases as well as external user databases can be used to lookup user credentials for CLI access. For more information about user authentication, please see section Chapter 8, *User Authentication*.

When accessing the CLI, the system will respond with the login prompt. Enter your username and press *Enter*, followed by your password and *Enter*. After a successful logon you will see the command prompt. If a welcome message has been set then it will be displayed directly after the logon:

```
Cmd>
```

**Tip**

For security reasons, it can be useful to disable or anonymize the CLI welcome message.

### Logging off from the CLI

After finishing working with the CLI, you should logout to avoid other people getting unauthorized access to the system. Log off by using the *exit* or the *logout* command.

## 2.1.4. Working with Configurations

### Configuration Objects

The system configuration is built up by *Configuration Objects*, where each object represents a configurable item of any kind. Examples of configuration objects are routing table entries, address book entries, service definitions, IP rules and so forth. Each configuration object has a number of properties that constitute the values of the object.

A configuration object has a well-defined type. The type defines the properties that are available for the configuration object, as well as the constraints for those properties. For instance, the *IP4Address* type is used for all configuration objects representing a named IPv4 address.

In the Clavister FineTune the configuration objects are organized into a tree-like structure based on the type of the object.

## Listing Configuration Objects

To find out what configuration objects exist, you can retrieve a listing of the objects.

### Example 2.1. Listing Configuration Objects

This example shows how to list all service objects.

#### *Clavister FineTune*

1. Select the **Services** section of the target system in the tree view of the Security Editor.
2. A grid listing all services will be presented.

## Displaying a Configuration Object

The most simple operation on a configuration object is just to show its contents, in other words the values of the object properties.

### Example 2.2. Displaying a Configuration Object

This example shows how to display the contents of a configuration object representing the *telnet* service.

#### *Clavister FineTune*

1. Select the **Services** section of the target system in the tree view of the Security Editor.
2. Select the **telnet** service object in the grid control.
3. Choose **Properties** from the context menu. The Service Properties dialog box will be displayed.

## Editing a Configuration Object

When you need to modify the behavior of CorePlus, you will most likely need to modify one or several configuration objects.



### ***Important***

*Changes to a configuration object will not be applied to a running system until you activate and commit the changes.*

### Example 2.3. Editing a Configuration Object

This example shows how to edit the *Comments* property of the *telnet* service.

#### *Clavister FineTune*

1. Select the **Services** section of the target system in the tree view of the Security Editor.
2. Select the **telnet** service object in the grid control.
3. Choose **Properties** from the context menu. The Service Properties dialog box will be displayed.
4. In the **Comment** textbox, enter your new comment.
5. Click the **OK**.

## Adding a Configuration Object

### Example 2.4. Adding a Configuration Object

This example shows how to add a new *IP4Address* object, here using the IP address 192.168.10.10, to the Address Book.

#### *Clavister FineTune*

1. Select the **Hosts & Networks** section of the target system in the tree view of the Security Editor.
2. Choose **New Host & Network** from the context menu. The **Host & Network Properties** dialog box will be displayed.
3. Specify a suitable name for the object, for instance myhost.
4. Choose **Host** in the **Type** control.
5. Enter 192.168.10.10 in the **IP Address** textbox.
6. Click **OK**.

## Deleting a Configuration Object

### Example 2.5. Deleting a Configuration Object

This example shows how to delete the newly added IP4Address object.

#### *Clavister FineTune*

1. Select the **Hosts & Networks** section of the target system in the tree view of the Security Editor.
2. Right-click on the row containing the **myhost** object
3. In the menu displayed, select **Delete**
4. Click **OK**.

## Activating and Committing a Configuration

When changes to a configuration have been made, the configuration has to be activated for those changes to have an impact on the running system. During the activation process, the new proposed configuration is validated and CorePlus will attempt to initialize affected subsystems with the new configuration data.



### **Committing IPsec Changes**

*The administrator should be aware that if any changes that effect the configurations of live IPsec tunnels are committed, then those live tunnels connections WILL BE TERMINATED and must be re-established.*

If the new configuration is validated, CorePlus will wait for a short period (30 seconds by default) during which a connection to the administrator must be re-established. If the connection could not be re-established the system will revert to using the previous configuration. This is a powerful fail-safe mechanism as it will, amongst others things, prevent you from locking yourself out of the gateway when using a remote system.

**Example 2.6. Activating and Committing a Configuration**

This example shows how to activate and commit a new configuration.

**Clavister FineTune**

1. Bring up the context menu for the target gateway
2. Choose **Check-in**
3. Enter a comment if required and confirm

**Note**

*All changes to a configuration can be ignored simply by not committing a changed configuration.*

## 2.1.5. NetCon

NetCon is a protocol, proprietary to Clavister, which is used for all communication between the Clavister FineTune and CorePlus. NetCon operates as a secure console command protocol, similar in approach to Secure Shell (SSH). It is a protocol that uses encryption based on key exchange and therefore provides a high level of security.

## 2.2. Events and Logging

### 2.2.1. Overview

The ability to log and analyze system activities is one of the most vital and fundamental features of a CorePlus system. Logging enables you not only to monitor system status and health, but also to audit the usage of your network as well as assisting you with debugging functionality.

CorePlus defines a number of *event messages*, which are generated as a result of corresponding system events. Examples of such events are establishment and teardown of connections, receiving malformed packets, dropping traffic according to filtering policies and so forth.

Whenever an event message is generated, it can be filtered and distributed to a variety of *Event Receivers*, including Syslog and SNMP Trap receivers. Up to eight event receivers can be defined, with each receiver having its own customizable event filter.

The sophisticated design of the event and logging mechanisms of CorePlus ensures that enabling logging is simple and straightforward, while it still allows a granular control of all the activities in the system for the more advanced deployments.

### 2.2.2. Event Messages

CorePlus defines several hundred events for which event messages can be generated. The events range from high-level, customizable, user events down to low-level and mandatory system events.

The *conn\_open* event, for instance, is a typical high-level event that generates an event message whenever a new connection is established, given that the matching security policy rule has defined that event messages should be generated for that connection.

An example of a low-level event would be the *startup\_normal* event, which generates a mandatory event message as soon as the system starts up.

All event messages have a common design, with attributes like category, severity, recommended actions and so forth. These attributes enables you to easily filter the event messages, either within CorePlus prior to sending them to an event receiver, or as part of the analysis taking place after logging and storing the messages on an external log server.



#### **Note**

*A list of all event messages can be found in the Log Reference Guide. That guide also describes the design of event messages, and explains the various attributes available.*

### 2.2.3. Event Message Distribution

To distribute and log the event messages generated, it's necessary to define one or more event receivers that specify *what* events to capture, and *where* to send them.

CorePlus can distribute event messages using the following standards and protocols:

<b>Syslog</b>	The de-facto standard for logging events from network devices. If you have other network devices logging to syslog hosts, you should consider using syslog from CorePlus as well to simplify your overall log administration.
<b>FWLog</b>	The Clavister proprietary format for logging event messages, the FWLog format has a high level of detail and is highly suitable for analyzing large amounts of log data.
<b>SNMP Traps</b>	Commonly used to collect and respond to critical alerts from net-

work devices.

#### NetCon Real-time Log

Event messages will always be distributed to any Real-time Log listeners that are connecting using the NetCon protocol.

### 2.2.3.1. Logging to Syslog Hosts

Syslog is a standardized protocol for sending log data to loghosts, although there is no standardized format of these log messages. The format used by CorePlus is well suited for automated processing, filtering and searching.

Although the exact format of each log entry depends on how your syslog recipient works, most are very much alike. The way in which logs are read is also dependent on how your syslog recipient works. Syslog daemons on UNIX servers usually log to text files, line by line.

Most syslog recipients preface each log entry with a timestamp and the IP address of the machine that sent the log data:

```
Feb 5 2000 09:45:23 gateway.ourcompany.com
```

This is followed by the text the sender has chosen to send.

```
Feb 5 2000 09:45:23 gateway.ourcompany.com EFW: DROP:
```

Subsequent text is dependent on the event that has occurred.

In order to facilitate automated processing of all messages, CorePlus writes all log data to a single line of text. All data following the initial text is presented in the format name=value. This enables automatic filters to easily find the values they are looking for without assuming that a specific piece of data is in a specific location in the log entry.

#### Example 2.7. Enable Logging to a Syslog Host

To enable logging of all events with a severity greater than or equal to Notice to a syslog server with IP address 195.11.22.55, follow the steps outlined below:

##### *Clavister FineTune*

1. Select the **Event Handling/Event Receivers** section of the target system in the tree view of the Security Editor.
2. Choose **New Event Receiver...** from the context menu. The **Event Receiver Properties** dialog box will be displayed.
3. Specify a suitable name for the event receiver, for instance my\_syslog.
4. Select **Syslog** in the **Type** dropdown list.
5. Enter 195.11.22.55 in the **IP Address** textbox.
6. Select an appropriate facility in the **Facility** dropdown list. The facility name is commonly used as a filter parameter in most syslog daemons.
7. Click **OK**.

The system will now be logging all events with a severity greater than or equal to Notice to the syslog server at 195.11.22.55.



#### **Note**

*The syslog server may have to be configured to receive log messages from CorePlus. Please see the documentation for your specific Syslog server software in order to correctly configure it.*

## 2.2.3.2. Logging to the Clavister Logger

The Clavister Logger uses the proprietary Clavister FWLog format for sending and storing log data. The FWLog format provides a high level of detail and is very useful for troubleshooting.

The Clavister Logger software runs as a service on a Microsoft Windows server, and is included as a utility. For more information about setting up and configuring the Clavister Logger please see the *Clavister FineTune Administration Guide*.

### Example 2.8. Enabling Logging to Clavister Loggers

To enable logging of all events with a severity greater than or equal to Notice to the Clavister Logger with IP address 195.11.22.55, follow the steps outlined below:

#### *Clavister FineTune*

1. Select the **Event Handling/Event Receivers** section of the target system in the tree view of the Security Editor.
2. Choose **New Event Receiver...** from the context menu. The **Event Receiver Properties** dialog box will be displayed.
3. Specify a suitable name for the event receiver, for instance my\_fwlog.
4. Select **FWLog** in the **Type** dropdown list.
5. Enter 195.11.22.55 in the **IP Address** textbox.
6. Click **OK**.

The system will now be logging all events with a severity greater than or equal to Notice to the Clavister FWLogger at 195.11.22.55.

## 2.2.3.3. SNMP Traps

### The SNMP protocol

Simple Network Management Protocol (SNMP) is a means for communicating between a Network Management System (NMS) and a managed device. SNMP defines 3 types of messages: a *Read* command for an NMS to examine a managed device, a *Write* command to alter the state of a managed device and a *Trap* which is used by managed devices to send messages asynchronously to an NMS about a change of state.

### SNMP Traps in CorePlus

CorePlus takes the concept of an SNMP Trap one step further by allowing *any* event message to be sent as an SNMP trap. This means that the administrator can set up SNMP Trap notification of events that you consider significant for the operation of a network.

The *Clavister-TRAP.MIB* file (included under the *SNMP* directory in the CorePlus distribution) defines the SNMP objects and datatypes that are used to describe an SNMP Trap received from CorePlus.

There is one generic trap object, , that is used for all traps. This object includes the following parameters:

- *System* - The system generating the trap
- *Severity* - Severity of the message
- *Category* - What CorePlus subsystem is reporting the problem

- *ID* - Unique identification within the category
- *Description* - A short textual description
- *Action* - What action is CorePlus taking

This information can be cross-referenced to the *Log Reference Guide*.



### Note

CorePlus sends SNMP Traps which are based on the SNMPv2c standard as defined by RFC1901, RFC1905 and RFC1906.

#### Example 2.9. Sending SNMP Traps to an SNMP Trap Receiver

To enable generation of SNMP traps for all events with a severity greater than or equal to Alert to an SNMP trap receiver with an IP address of 195.11.22.55, follow the steps outlined below:

##### Clavister FineTune

1. Select the **Event Handling/Event Receivers** section of the target system in the tree view of the Security Editor.
2. Choose **New Event Receiver...** from the context menu. The **Event Receiver Properties** dialog box will be displayed.
3. Specify a suitable name for the event receiver, for instance my\_snmp.
4. Select **SNMP2c** in the **Type** dropdown list.
5. Enter 195.11.22.55 in the **IP Address** textbox.
6. Enter an SNMP community string (if needed by the trap receiver) in the **Community String** textbox.
7. Click **OK**.

The system will now be sending SNMP traps for all events with a severity greater than or equal to Alert to an SNMP trap receiver at 195.11.22.55.

## 2.2.4. Customizing Event Message Distribution

As previously described, event messages gets distributed to the event message receivers based on their default severity as well as the severity filters configured for each receiver. The filters are pre-defined to best match the usage of each receiver type. For instance, a Syslog receiver type accepts all event messages except debug, while an SNMP Trap Receiver only forwards event messages with a severity of alert or emergency.

In some situations however, this filtering of event messages is not sufficient. In some networks, you might experience that some event messages are too frequent and you would like to disable logging of those. In other scenarios there might be a need to modify the severity of a certain event to better suit the setup of your event and alarm management system.

For these reasons, CorePlus employs a method to customize in detail what event messages should be included or excluded and also what severity a certain event should have when distributed to an event message receiver.

Each event message receiver contains an *Exception List* where you can enter exceptions from the severity filter. For instance, consider an SNMP Trap receiver with default severity filters (i.e. accepts all event messages with a severity of **Alert** or **Emergency**). In addition, the following exception list has been defined:

Include/Exclude	Event Message	New Severity	Comment
Include	01200001	Unchanged	HA peer_gone



<b>Include/Exclude</b>	<b>Event Message</b>	<b>New Severity</b>	<b>Comment</b>
Include	01200011	Unchanged	HA peer_alive

With this exception list, SNMP traps are also sent for the peer\_gone and peer\_alive event messages even though they have a default severity of Notice.

## 2.3. RADIUS Accounting

### 2.3.1. Overview

Within a network environment containing large numbers of users, it is advantageous to have one or a cluster of central servers that maintain user account information and are responsible for authentication and authorization tasks. The central database residing on the dedicated server(s) contains all user credentials as well as details of connections, significantly reducing administration complexity. The Remote Authentication Dial-in User Service (RADIUS) is an Authentication, Authorization and Accounting (AAA) protocol widely used to implement this approach and is used by CorePlus to implement user accounting.

The RADIUS protocol is based on a client/server architecture. The Clavister Security Gateway acts as the client of the RADIUS server, creating and sending requests to a dedicated server(s). In RADIUS terminology the gateway acts as the Network Access Server (NAS). For user authentication, the RADIUS server receives the requests, verifies the user's information by consulting its database, and returns either an "ACCEPT" or "REJECT" decision to the requested client. In RFC2866, RADIUS was extended to handle the delivery of accounting information and this is the standard followed by CorePlus for user accounting. The benefits of having centralized servers are thus extended to user connection accounting. (For details of the usage of RADIUS for CorePlus authentication see Section 8.2, "Authentication Components").

### 2.3.2. RADIUS Accounting messages

Statistics, such as number of bytes sent and received, and number of packets sent and received are updated and stored throughout RADIUS sessions. All statistics are updated for an authenticated user whenever a connection related to an authenticated user is closed.

When a new client session is started by a user establishing a new connection through the Clavister Security Gateway, CorePlus sends an *AccountingRequest* **START** message to a nominated RADIUS server, to record the start of the new session. User account information is also delivered to the RADIUS server. The server will send back an *AccountingResponse* message to CorePlus, acknowledging that the message has been received. The diagram below illustrates the message interaction.

When a user is no longer authenticated, for example, after the user logs out or the session time expires, an *AccountingRequest* **STOP** message is sent by CorePlus containing the relevant session statistics. The information included in these statistics is user configurable. The contents of the **START** and **STOP** messages are described in detail below:

#### START Message Parameters

Parameters included in **START** messages sent by CorePlus are:

- **Type** - Marks this *AccountingRequest* as signaling the beginning of the service (START).
- **ID** - A unique identifier to enable matching of an *AccountingRequest* with *Acct-Status-Type* set to STOP.
- **User Name** - The user name of the authenticated user.
- **NAS IP Address** - The IP address of the Clavister Security Gateway.
- **NAS Port** - The port of the NAS on which the user was authenticated (this is a physical port and not a TCP or UDP port).
- **User IP Address** - The IP address of the authenticated user. This is sent only if specified on the authentication server.
- **How Authenticated** - How the user was authenticated. This is set to either RADIUS if the user

was authenticated via RADIUS, or LOCAL if the user was authenticated via a local user database.

- **Delay Time** - The time delay (in seconds) since the AccountingRequest packet was sent and the authentication acknowledgement was received. This can be subtracted from the time of arrival on the server to find the approximate time of the event generating this AccountingRequest. Note that this does not reflect network delays. The first attempt will have this parameter set to 0.
- **Timestamp** - The number of seconds since 1970-01-01. Used to set a timestamp when this packet was sent from CorePlus.

### STOP Message Parameters

Parameters included in **STOP** messages sent by CorePlus are:

- **Type** - Marks this accounting request as signalling the end of a session (STOP).
- **ID** - An identifier matching a previously sent AccountingRequest packet, with Acct-Status-Type set to START.
- **User Name** - The user name of the authenticated user.
- **NAS IP Address** - The IP address of the Clavister Security Gateway.
- **NAS Port** - The port on the NAS on which the user was authenticated. (this is a physical port and not a TCP or UDP port).
- **User IP Address** - The IP address of the authenticated user. This is sent only if specified on the authentication server
- **Input Bytes** - The number of bytes received by the user. (\*)
- **Output Bytes** - The number of bytes sent by the user. (\*)
- **Input Packets** - The number of packets received by the user. (\*)
- **Output Packets** - The number of packets sent by the user. (\*)
- **Session Time** - The number of seconds this session lasted. (\*)
- **Termination Cause** - The reason why the session was terminated.
- **How Authenticated** - How the user was authenticated. This is set to either *RADIUS* if the user was authenticated via RADIUS, or *LOCAL* if the user was authenticated via a local user database.
- **Delay Time** - See the above comment about this parameter.
- **Timestamp** - The number of seconds since 1970-01-01. Used to set a timestamp when this packet was sent from the Clavister Security Gateway. In addition to this, two more attributes are possibly sent:
- **Input Gigawords** - Indicates how many times the Input Bytes counter has wrapped. This is only sent if Input Bytes has wrapped, and if the Input Bytes attribute is sent.
- **Output Gigawords** - Indicates how many times the Output Bytes counter has wrapped. This is only sent if Output Bytes has wrapped, and if the Output Bytes attribute is sent.

**Note**

The (\*) symbol in the above list indicates that the sending of the parameter is user configurable.

## 2.3.3. Interim Accounting Messages

In addition to **START** and **STOP** messages CorePlus can optionally periodically send *Interim Accounting Messages* to update the accounting server with the current status of an authenticated user. An Interim Accounting Message can be seen as a snapshot of the network resources that an authenticated user has used up until a given point. With this feature, the RADIUS server can track how many bytes and packets an authenticated user has sent and received up until the point when the last message was sent.

An Interim Accounting Message contains the current values of the statistics for an authenticated user. It contains more or less the same parameters as found in an AccountingRequest Stop message, except that the *Acct-Terminate-Cause* is not included (as the user has not disconnected yet).

The frequency of Interim Accounting Messages can be specified either on the authentication server, or in CorePlus. Switching on the setting in CorePlus will override the setting on the accounting server.

## 2.3.4. Activating RADIUS Accounting

In order to activate RADIUS accounting a number of steps must be followed:

- The RADIUS accounting server must be specified.
- A user authentication object must have a rule associated with it where a RADIUS server is specified.

Some important points should be noted about activation:

- RADIUS Accounting will not function where a connection is subject to a **FwdFast** rule in the IP rule-set.
- The same RADIUS server does not need to handle both authentication and accounting; one server can be responsible for authentication while another is responsible for accounting tasks.
- Multiple RADIUS servers can be configured in CorePlus to deal with the event when the primary server is unreachable.

## 2.3.5. RADIUS Accounting Security

Communication between CorePlus and any RADIUS accounting server is protected by the use of a shared secret. This secret is never sent over the network but instead a 16 byte long *Authenticator code* is calculated using a one way MD5 hash function and this is used to authenticate accounting messages.

The shared secret is case sensitive, can contain up to 100 characters, and must be typed exactly the same for CorePlus and for the RADIUS server.

Messages are sent using the UDP protocol and the default port number used is 1813 although this is user configurable.

## 2.3.6. RADIUS Accounting and High Availability

In an HA cluster, accounting information is synched between the active and passive Clavister Security Gateways. This means that accounting information is automatically updated on both cluster members whenever a connection is closed. Two special accounting events are also used by the active unit to keep the passive unit synchronized:

- An **AccountingStart** event is sent to the inactive member in an HA setup whenever a response has been received from the accounting server. This specifies that accounting information should be stored for a specific authenticated user.
- A lack of accounting information synching could occur if an active unit has an authenticated user for whom the associated connection times out before it is synched to the inactive unit. To get around this problem, a special **AccountingUpdate** event is sent to the passive unit on a timeout and this contains the most recent accounting information for connections.

## 2.3.7. Handling Unresponsive Servers

A question arises in the case of a client that sends an *AccountingRequest START* packet which the RADIUS server never replies to. CorePlus will re-send the request after the user-specified number of seconds. This will however mean that a user will still have authenticated access while CorePlus is trying to contact to the accounting server.

Only after CorePlus has made three attempts to reach the server will it conclude that the accounting server is unreachable. The administrator can use the CorePlus advanced setting **AllowAuthIfNoAccountingResponse** to determine how this situation is handled. If this setting is enabled then an already authenticated user's session will be unaffected. If it is not enabled, any effected user will automatically be logged out even if they have already been authenticated.

## 2.3.8. Accounting and System Shutdowns

In the case that the client for some reason fails to send a RADIUS *AccountingRequest STOP* packet (due to, for example, the accounting server will never be able to update its user statistics, but will most likely believe that the session is still active and this situation should be avoided.

In the case that the Clavister Security Gateway administrator issues a shutdown command while authenticated users are still online, the *AccountingRequest STOP* packet will potentially never be sent. To avoid this, CorePlus has the advanced setting **LogOutAccUsersAtShutdown**. This setting allows the administrator to explicitly specify that CorePlus must first send a **STOP** message for any authenticated users to any configured RADIUS servers before commencing with the shutdown.

## 2.3.9. Limitations with NAT'ed Networks

The User Authentication module in CorePlus is based on the user's IP address. Problems can therefore occur with users who have the same IP address.

This can happen, for instance, when several users are behind the same network and which uses NAT to allow network access. This means that as soon as one user is authenticated, all users from the same network are also authenticated. CorePlus RADIUS Accounting will therefore gather statistics for all the users on the network together as though they were one user instead of individuals.

## 2.4. Monitoring

### 2.4.1. Real-time Monitoring Counters

The following counters are available through real-time monitoring of the Clavister Security Gateway:

#### Throughput statistics

**CPU** – the load in percent on the gateway CPU.

**Forwarded bps** - The number of bits forwarded through the gateway per second.

**Forwarded pps** - The number of packets forwarded through the gateway per second.

**Buf use** – Percentage of gateway packet buffers used.

**Conns** – The number of connections opened via Allow or NAT rules. No connections are opened for traffic allowed via FwdFast rules; such traffic is not included in this graph.

#### Rule statistics

Each rule in the rule-set has a counter associated with it and having the same name as the rule. This counter indicates the number of matches that have taken place for each rule.

#### Interface statistics

**Rx/Tx Ring counters** - Some drivers support plotting of FIFO-errors, Saturation, flooding and more values depending on driver.

**Pps counters** – The number of packets received, sent and summed together.

**Bbs** – The number of bits received, sent and summed together.

**Drops** – The number of packets received by this interface that were dropped due to ruleset decisions or failed packet consistency checks.

**IP errors** – The number of packets received by this interface that were mutilated so badly that they would have had difficulty passing through a router to reach to the Clavister Security Gateway. They are therefore unlikely to be the result of an attack.

**Send Fails** – The number of packets that could not be sent, either due to internal resource starvation caused by heavy loading or hardware problems or congested half-duplex connections.

**Frag received** – The number of IP packet fragments received by this interface.

**Frag reas** – the number of complete packets successfully reassembled from the fragments received.

**Frag reas fail** – The number of packets that could not be reassembled, either due to resource starvation, illegal fragmentation, or just packet loss.

#### DHCP Server statistics

**Total Rejected requests** – Total number of rejected packets (all rules).

##### Per Rule statistics

**Usage** – Number of used IPs in the pool.

**Usage (%)** – Above value calculated in %.

**Active Clients** – Number of currently active clients (BOUND).

**Active Clients (%)** – Above value calculated in %

**Reject requests** – Number of rejected requests.

**Total number of leases** – Total number of leases in the pool.

## Per-Pipe statistics

### Total Pipe statistics

**Num users** – The current number of users, as defined by the grouping settings of each pipe, being tracked in the pipes system. Note that this value corresponds to the number of users active in each time slice of 1/20th of a second, and not to the number of users having "open" connections.

### Per-Pipe statistics

**Current Bps** – The current throughput of the pipe, measured in bits per second, per precedence and as a sum of all precedences.

**Current Pps** – The current throughput of the pipe, measured in bits per second, per precedence and as a sum of all precedences.

**Resrvd Bps** – The current bandwidth allocated to each precedence; lower precedences are not allowed to use this bandwidth. Note that there is no reserved bandwidth for precedence 0, as it is simply given what is left of the total limit after all higher precedence reservations are subtracted.

**Dyn limit Bps** – The current bandwidth limit applied to the respective precedences. This is related to the Reserved Bps graph, but is usually higher, as it shows how much bandwidth is left after higher precedence reservations have been subtracted from the total limit.

**Delayed Packets** – The number of times packets have been delayed as a result of a pipe, precedence, or pipe user having used up its allotted bandwidth. Note that one single packet may be delayed several times; if a pipe is really full, this count may exceed the number of packets actually passing through the pipe.

**Dropped Packets** – The number of packets dropped. Packets are dropped when CorePlus is running out of packet buffers. This occurs when excessive amounts of packets need to be queued for later delivery. The packet dropped is always the one that has been queued the longest time globally, which means that the connection suffering from packet loss will be the one most overloading the system .

**Dyn User Limit Bps** – The current bandwidth limit per user of the pipe. If dynamic bandwidth balancing is enabled, this value may be lower than the configured per-user limits.

## VPN statistics

**Active SAs** - Current active number of SAs in use.

### VPN Interface statistics

**Pps counters** – The number of packets received and sent, added together.

**Bps counters** – The number of bits received and sent, added together.

**Dropped** – The number of packets received by this interface that were dropped due to ruleset decisions or failed packet consistency checks.

**IP Errors** – The number of packets received by this interface that were mutilated so badly that they would have had difficulty passing through a router to reach to the Clavister Security Gateway. They

are therefore unlikely to be the result of an attack.

**Send Fails** – The number of packets that could not be sent, either due to internal resource starvation caused by heavy load, hardware problems, or congested half-duplex connections.

**Fragments Received** – The number of IP packet fragments received by this interface.

**Frag reass OK** – The number of complete packets successfully reassembled from the fragments received.

**Frag reass fail** – the number of packets that could not be reassembled, either due to resource starvation, illegal fragmentation, or just packet loss.

## User Authentication statistics

**UA Rule Use** – The usage counter for a specific User Authentication rule in the configuration can be plotted by selecting the appropriate rule in the UA Rule Use section.

**HTTPAuth** – Number of authenticated users.

## DHCPRelay statistics

**Total active relayed clients** - Number of active relays in the Clavister Security Gateway.

**Ongoing transactions** - DHCP transactions in the gateway.

**Total packets rejected** - Number of packets rejected by the DHCPRelay.

### DHCPRelay Rule statistics

**Hits** - Number of times the specific DHCPRelay rule has been used.

**Active relayed clients** - Number of active relays that uses this specific rule.

**Client packets rejected** - Number of packets rejected from clients using this rule.

**Server packets rejected** - Number of packets rejected from the server using this rule.

## ALG statistics

**Total ALG sessions** - Total number of ALG sessions.

**Total connections** - Total number of connections.

**Total TCP Streams** - Total number of TCP streams.

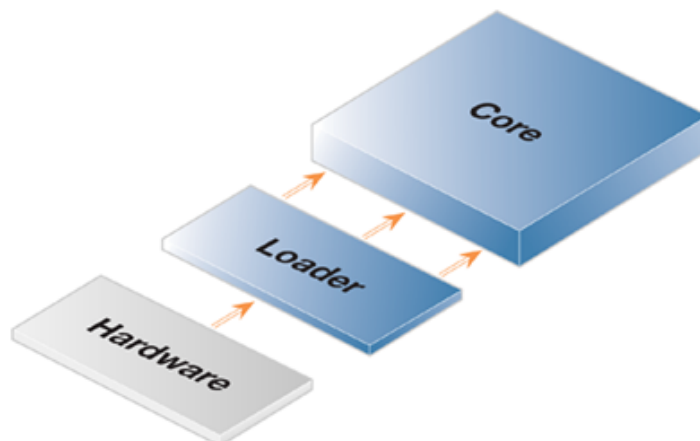


## 2.5. Maintenance

### 2.5.1. Firmware Upgrades

Clavister Security Gateways are driven and controlled by the CorePlus firmware and this consists of two major components: the CorePlus Loader and the CorePlus Core.

**Figure 2.1. CorePlus Firmware Structure**



#### 2.5.1.1. CorePlus Core

The CorePlus Core is the major building block of the system, and contains all the functional mechanisms, including packet forwarding engine, control plane tasks and so forth.

Whenever new functionality is added to CorePlus, or when defects have been found and corrected, a new CorePlus Core is produced. The new Core is packaged, digitally signed and made available for download from Clavister.

There are two types of CorePlus Core upgrades:

- Minor upgrades, for instance from version 8.70 to 8.71, include bug fixes and minor feature enhancements and are freely available to all customers.
- Major upgrades, for instance from version 8.70 to 8.80, are only available for customers that have a valid software subscription service.



**Note**

*All open connections through the system will be dropped when the system is upgraded with a new Core.*

#### **Example 2.10. Upgrading the CorePlus Core**

**Clavister FineTune**

1. Go to **Upgrade > CorePlus Core** from the **Action > Communication** menu. The **Upgrade CorePlus Core Wizard** will be displayed.

2. Select the Core that should be uploaded to the system and click the **Next** button.
3. If the required Core is not shown in the list, the **Browse...** button can be used to browse the file system for a Core in another location.
4. When the Core has been uploaded, the system will perform a shutdown and then restart with the new Core.

### 2.5.1.2. CorePlus Loader

The CorePlus Loader can be viewed as a hardware abstraction layer that contains all the mechanisms for interfacing to the system hardware, as well as to low-level kernel functionality. Upgrading the CorePlus loader is rarely needed and when performed, it will be in conjunction with a new CorePlus version and will be documented in the CorePlus release notes.

## 2.5.2. Reset to Factory Defaults

It is possible to apply the original defaults that existed when the Clavister Security Gateway was purchased. These defaults can be applied only to the current configuration or to the entire hardware unit. The latter option essentially restores the unit to the state it was in when it left the factory.

### Reset alternative via the Serial Console

Connect the serial cable and connect with a console using a terminal emulator software product. If Microsoft Windows is being used, "HyperTerminal" can be used. Reset the gateway. Press a key when the "Press any key to abort and load boot menu" message appears at the console. When the boot menu appears, select the desired option then confirm and wait for the process to complete.



#### **Warning**

*DO NOT ABORT THE RESET TO FACTORY DEFAULTS PROCESS. If aborted the gateway can cease to function properly.*

## 2.5.3. Configuration Backup and Restore

The configuration of Clavister Security Gateways can be backed up or restored on demand. This could, for instance, be used to recall the "last known good" configuration when experimenting with different configuration setups.



#### **Note**

*Backups include only static information in the gateway configuration. Dynamic information such as the DHCP server lease database will not be backed up.*

## 2.5.4. Auto-Update Mechanism

A number of the CorePlus security features rely on external servers for automatic updates and content filtering. The Intrusion Prevention and Detection system and Anti-Virus modules require access to updated signature databases in order to provide protection against the latest threats.

To facilitate the Auto-Update feature Clavister maintains a global infrastructure of servers providing update services for Clavister Security Gateways. To ensure availability and low response times, CorePlus employs a mechanism for automatically selecting the most appropriate server to supply updates.

For more details on these features see the following sections:

- Section 6.3, "Intrusion Detection and Prevention"

- Section 6.4, “Anti-Virus”
- Section 6.5, “Web Content Filtering”
- Appendix A, *Subscribing to Security Updates*



---

# Chapter 3. Fundamentals

This chapter describes the fundamental logical objects upon which CorePlus is built. These logical objects include such things as addresses, services and schedules. In addition, this chapter explains how the various supported interfaces work, it outlines how policies are constructed and how basic system settings are configured.

- The Address Book, page 32
- Services, page 36
- Interfaces, page 40
- ARP, page 48
- The IP Rule-Set, page 53
- Schedules, page 56
- X.509 Certificates, page 58
- System Settings, page 60

## 3.1. The Address Book

### 3.1.1. Overview

The Address Book contains named objects representing various types of addresses, including IP addresses, networks and Ethernet MAC addresses.

Using Address Book objects has three distinct benefits; it increases readability, reduces the danger of entering incorrect network addresses, and makes it easier to change addresses. By using objects instead of numerical addresses, you only need to make changes in a single location, rather than in each configuration section where the address appears.

### 3.1.2. IP Addresses

*IP Address* objects are used to define symbolic names for various types of IP addresses. Depending on how the address is specified, an IP Address object can represent either a host (a single IP address), a network, a range of IP addresses or even a DNS name.

In addition, IP Address objects can be used for specifying user credentials later used by the various user authentication subsystems. For more information on this, see Chapter 8, *User Authentication*.

The following list presents the various types of addresses an IP Address object can hold, along with what format that is used to represent that specific type:

**Host**                      A single host is represented simply by its IP address.  
For example: *192.168.0.14*

**IP Network**              An IP Network is represented using CIDR (Classless Inter Domain Routing) form. CIDR uses a forward slash and a digit (0-32) to denote the size of the network (netmask). */24* corresponds to a class C net with 256 addresses (netmask 255.255.255.0), */27* corresponds to a 32 address net (netmask 255.255.255.224) and so forth. The numbers 0-32 correspond to the number of binary ones in the netmask.  
For example: *192.168.0.0/24*

**IP Range** A range of IP addresses is represented on the form *a.b.c.d - e.f.g.h*. Please note that ranges are not limited to netmask boundaries; they may include any span of IP addresses.

For example: *192.168.0.10-192.168.0.15* represents six hosts in consecutive order.

**DNS Address** A DNS address is represented by the form *dns:domainname*. For example: *dns:www.sunet.se*

### Example 3.1. Adding an IP Host

This example adds the IP host *wwwsrv1* with IP address *192.168.10.16* to the Address Book:

#### **Clavister FineTune**

1. Go to **Hosts & Networks**
2. Choose **New Host & Network...** from the context menu and the **Host & Network Properties** dialog box will be displayed
3. Specify a suitable name for the event receiver, in this case *wwwsrv1*
4. Choose **Host** in the **Type** control
5. Enter *192.168.10.16* in the **IP Address** textbox
6. Click **OK**

### Example 3.2. Adding an IP Network

This example adds an IP network named *wwwsrvnet* with address *192.168.10.0/24* to the Address Book:

#### **Clavister FineTune**

1. Select the **Hosts & Networks** section of the target system in the tree view of the Security Editor.
2. Choose **New Host & Network...** from the context menu, the **Host & Network Properties** dialog box will be displayed.
3. Specify a suitable name for the event receiver, in this case *wwwsrvnet*.
4. Choose **Network** in the **Type** control.
5. Enter *192.168.10.0/24* in the **IP Address** textbox.
6. Click **OK**.

### Example 3.3. Adding an IP Range

This example adds a range of IP addresses from *192.168.10.16* to *192.168.10.21* and names the range *wwwservers*:

#### **Clavister FineTune**

1. Select the **Hosts & Networks** section of the target system in the tree view of the Security Editor.
2. Choose **New Host & Network...** from the context menu, the **Host & Network Properties** dialog box will be displayed.

3. Specify a suitable name for the event receiver, in this case `wwwsrvnet`.
4. Choose **Range** in the **Type** control.
5. Enter `192.168.10.16` in the **IP Low** textbox.
6. Enter `192.168.10.21` in the **IP High** textbox.
7. Click **OK**.

#### Example 3.4. Deleting an Address Object

To delete an object named `wwwsrv1` in the Address Book, do the following:

##### *Clavister FineTune*

1. Select the **Hosts & Networks** section of the target system in the tree view of the Security Editor.
2. Select and right-click the address object `wwwsrv1` in the grid.
3. Choose **Delete** in the menu.
4. Click **Yes** in the popup.

### 3.1.3. Ethernet Addresses

*Ethernet Address* objects are used to define symbolic names for Ethernet addresses (also known as MAC addresses). This is useful, for instance, when populating the ARP table with static ARP entries, or for other parts of the configuration where symbolic names are preferred over numerical Ethernet addresses.

When specifying an Ethernet address the format *aa-bb-cc-dd-ee-ff* should be used. Ethernet addresses are also displayed using this format.

### 3.1.4. Address Groups

Address objects can be grouped in order to simplify configuration. Consider a number of public servers that should be accessible from the Internet. The servers have IP addresses that are not in a sequence, and can therefore not be referenced to as a single IP range. Consequently, individual IP Address objects have to be created for each server.

Instead of having to cope with the burden of creating and maintaining separate filtering policies allowing traffic to each server, an *Address Group* named, for instance, *Webservers*, can be created with the web server hosts as group members. Now, a single policy can be used with this group, thereby greatly reducing the administrative workload.

Address Group objects are not restricted to contain members of the same subtype. In other words, IP host objects can be teamed up with IP ranges, IP networks with DNS names and so forth. All addresses of all group members are combined, effectively resulting in a union of the addresses. As an example, a group containing two IP ranges, one with addresses *192.168.0.10 - 192.168.0.15* and the other with addresses *192.168.0.14 - 192.168.0.19*, will result in a single IP range with addresses *192.168.0.10 - 192.168.0.19*.

Keep in mind however that for obvious reasons, IP address objects can not be combined with Ethernet addresses.

### 3.1.5. Auto-Generated Address Objects

To simplify the configuration, several address objects are automatically generated when the system is run for the first time. These objects are being used by other parts of the configuration already from start.

The following address objects are auto-generated:

#### **Interface Addresses**

For each Ethernet interface in the system, three IP Address objects are pre-defined; one object for the IP address of the actual interface, one for the broadcast address, and one object representing the local network for that interface.

Interface IP address objects are named *ip\_interfacename*, broadcast objects are named *br\_interfacename* and network objects are named *interfacenamenet*. As an example, an interface named *lan* will have an associated interface IP object named *ip\_lan*, a broadcast object named *br\_lan* and a network object named *lannet*.

#### **Default Gateway**

An IP Address object named *defaultgw* is auto-generated and represents the default gateway of the system. The *defaultgw* object is used primarily by the routing table, but is also used by the DHCP client subsystem to store gateway address information acquired from a DHCP server. The object will have an empty IP address (0.0.0.0), and you need to define the correct IP address for your default gateway unless you are using DHCP.

#### **all-nets**

The *all-nets* IP address object is initialized to the address 0.0.0.0/0, thus representing all possible IP addresses. This object is used extensively throughout the configuration.



## 3.2. Services

### 3.2.1. Overview

A **Service** object is a reference to a specific IP protocol with associated parameters. A Service definition is usually based on one of the major transport protocols such as TCP or UDP, with the associated port number(s). The HTTP service, for instance, is defined as using the TCP protocol with associated port 80.

Service objects are in no way restricted to TCP or UDP; they can be used to define ICMP messages, as well as any user-definable IP protocol.

Services are simple objects in that they cannot carry out any action in the system on their own. Instead, Service objects are used frequently by the various system policies. For instance, a rule in the IP Rule-set can use a Service object as a filter to decide whether or not to allow certain traffic through the gateway. For more information on how service objects are being used in policies, please see Section 3.5, “The IP Rule-Set”.

A great number of Service objects comes pre-defined with the CorePlus. These include common services such as HTTP, FTP, Telnet and SSH. These pre-defined services can be used and also modified just like user-defined Services. However it is advisable not to make any changes to pre-defined services, but instead create new ones with the desired parameters.

#### Example 3.5. Listing the Available Services

To produce a listing of the available services in the system:

##### *Clavister FineTune*

1. Select the **Services** section of the target system in the tree view of the Security Editor.
2. A grid listing all services will be presented.

#### Example 3.6. Viewing a Specific Service

To view a specific service in the system:

##### *Clavister FineTune*

1. Go to the **Services** section of the target system in the tree view of the Security Editor.
2. Select the specific service object in the grid control.
3. Choose **Properties** from the context menu. The Service Properties dialog box will be displayed.

### 3.2.2. TCP and UDP Based Services

Most applications are using TCP and/or UDP as transport protocol for transferring application data over IP networks.

TCP (Transmission Control Protocol) is a connection-oriented protocol that, among other things, includes mechanisms for reliable transmission of data. TCP is used by many common applications, such as HTTP, FTP and SMTP, where error-free transfers are mandatory.

For other types of applications where, for instance, performance is of great importance, such as streaming audio and video services, UDP (User Datagram Protocol) is the preferred protocol. UDP is connection-less, provides very few error recovery services, and give thereby much lower over-

head traffic than when using TCP. For this reason, UDP is used for non-streaming services as well, and it is common in those cases that the applications themselves provide the error recovery mechanisms.

To define a TCP or UDP service in the Clavister Security Gateway, a *TCP/UDP Service* object is used. This type of object contains, apart from a unique name describing the service, also information on what protocol (TCP, UDP or both) and what source and destination ports are applicable for the service.

Port numbers can be specified in several ways:

#### Single Port

For many services, a single destination port is sufficient. HTTP, for instance, uses destination port 80 in most cases, SMTP uses port 25 and so forth. For this type of services, the single port number is simply specified in the TCP/UDP Service object.

#### Port Ranges

Some services use a range of destination ports. As an example, the NetBIOS protocol used by Microsoft Windows uses destination ports 137 to 139. To define a range of ports in a TCP/UDP Service object, the format *mmm-nnn* is used. A port range is inclusive, meaning that a range specified as 137-139 covers ports 137, 138 and 139.

#### Multiple Ports and Port Ranges

Multiple ranges or individual ports may also be entered, separated by commas. This provides the possibility to cover a wide range of ports using only a single TCP/UDP Service object. For instance, all Microsoft Windows networking can be covered using a port definition specified as *135-139,445*. HTTP and Secure HTTP (HTTPS) can be covered by stating destination ports *80,443*.



#### Tip

*The above methods of specifying port numbers are not used just for destination ports. Source port definitions can follow the same conventions, although it is most usual that the source ports are left as their default values, namely 0-65535, which matches all possible source ports.*

#### Example 3.7. Adding a TCP/UDP Service

This example shows how to add a TCP/UDP Service, using destination port 3306, which is used by MySQL:

##### **Clavister FineTune**

1. Select the **Local Objects > Services** section of the target system in the tree view of the Security Editor.
2. Choose **New Service...** from the context menu, the **Service Properties** dialog box will be displayed.
3. Specify a suitable name for the service, for instance MySQL.
4. Choose **TCP** in the **Type** control.
5. Under the **TCP/UDP Parameters**, select **Destination Port** and enter 3306.
6. Click **OK**.

Apart from protocol and port information, TCP/UDP Service objects also contain several other parameters that are being described in more detail in other sections of this users guide:

<b>SYN Flood Protection</b>	A TCP based service can be configured to enable protection against <i>SYN Flood</i> attacks.
<b>Passing ICMP Errors</b>	If an attempt to open a TCP connection is made by a user application behind the Clavister Security Gateway and the remote server is not in operation, an ICMP error message is returned as the response. These ICMP errors can either be ignored or allowed to pass through, back to the requesting application.
<b>Application Layer Gateway</b>	A TCP/UDP Service can be linked to an <i>Application Layer Gateway</i> to enable deeper inspection of certain protocols. For more information, please see Section 6.2, “Application Layer Gateways”.

### 3.2.3. ICMP Services

Internet Control Message Protocol (ICMP), is a protocol integrated with IP for error reporting and transmitting control information. The PING service, for example, uses ICMP to test an Internet connectivity.

ICMP messages is delivered in IP packets, and includes a *Message Type* that specifies the type, that is, the format of the ICMP message, and a *Code* that is used to further qualify the message. For example, the message type *Destination Unreachable*, uses the Code parameter to specify the exact reason for the error.

The ICMP message types that can be configured in CorePlus are listed as follows:

- Echo Request: sent by PING to a destination in order to check connectivity.
- Destination Unreachable: the source is told that a problem has occurred when delivering a packet. There are codes from 0 to 5 for this type:
  - Code 0: Net Unreachable
  - Code 1: Host Unreachable
  - Code 2: Protocol Unreachable
  - Code 3: Port Unreachable
  - Code 4: Cannot Fragment
  - Code 5: Source Route Failed
- Redirect: the source is told that there is a better route for a particular packet. Codes assigned are as follows:
  - Code 0: Redirect datagrams for the network
  - Code 1: Redirect datagrams for the host
  - Code 2: Redirect datagrams for the Type of Service and the network
  - Code 3: Redirect datagrams for the Type of Service and the host
- Parameter Problem: identifies an incorrect parameter on the datagram.
- Echo Reply: the reply from the destination which is sent as a result of the Echo Request.
- Source Quenching: the source is sending data too fast for the receiver, the buffer has filled up.

- Time Exceeded: the packet has been discarded as it has taken too long to be delivered.

## 3.2.4. Custom IP Protocol Services

Services that run over IP and perform application/transport layer functions can be uniquely identified by *IP protocol numbers*. IP can carry data for a number of different protocols. These protocols are each identified by a unique IP protocol number specified in a field of the IP header, for example, ICMP, IGMP, and EGP have protocol numbers 1, 2, and 8 respectively.

CorePlus supports these types of IP protocols by the concept of *Custom IP Protocol Services*. Basically, a Custom IP Protocol service is a service definition giving a name to an IP protocol number. Some of the common IP protocols, such as IGMP, are already pre-defined in the system configuration.

Similar to the TCP/UDP port ranges described previously, a range of IP protocol numbers can be used to specify multiple applications for one service.



### **Note**

*The currently assigned IP protocol numbers and references are published by the Internet Assigned Numbers Authority (IANA) and can be found at <http://www.iana.org/assignments/protocol-numbers>*

### **Example 3.8. Adding a IP Protocol Service**

This example shows how to add an IP Protocol Service, with the Virtual Router Redundancy Protocol.

#### **Clavister FineTune**

1. Select the **Local Objects > Services** section of the target system in the tree view of the Security Editor.
2. Choose **New Service...** from the context menu, the **Service Properties** dialog box will be displayed.
3. Specify a suitable name for the service, for instance VRRP.
4. Choose **IPProto** in the **Type** control.
5. Under the **IP Protocol Parameters**, enter 112.
6. Click **OK**.

## 3.3. Interfaces

### 3.3.1. Overview

An **Interface** is one of the most important logical building blocks in CorePlus. All network traffic that passes through or gets terminated in the system is done so through one or several interfaces.

An interface can be seen as a doorway for network traffic to or from the system. Thus, when traffic enters the system through an interface, that interface would be referred to as the *receiving* interface (or sometimes *ingress* or *incoming* interface). Consequently, when traffic is leaving the system, the interface used to send the traffic is referred to as the *sending* interface (or sometimes *egress* interface).

CorePlus supports a number of interface types, which can be divided into the following four major groups:

#### Physical Interfaces

Each *physical interface* represents a physical port in a CorePlus-based product. Thus, all network traffic that originates from or is terminated in the system will eventually pass through any of the physical interfaces.

CorePlus currently supports *Ethernet* as the only physical interface type. For more information about Ethernet interfaces, please see Section 3.3.2, “Ethernet”.

#### Physical Sub-Interfaces

Some interfaces require a binding to an underlying physical interface in order to transfer data. This group of interfaces is called *Physical Sub-Interfaces*.

CorePlus has support for two types of physical sub-interfaces:

- *Virtual LAN* (VLAN) interfaces as specified by IEEE 802.1Q. When routing IP packets over a Virtual LAN interface, they will be encapsulated in VLAN-tagged Ethernet frames. For more information about Virtual LAN interfaces, please see Section 3.3.3, “Virtual LAN”.
- *PPPoE* (PPP-over-Ethernet) interfaces for connections to PPPoE servers. For more information about PPPoE, please see Section 3.3.4, “PPPoE”.

#### Tunnel Interfaces

*Tunnel interfaces* are used when network traffic is being tunneled between the system and another tunnel end-point in the network, before it gets routed to its final destination.

To accomplish tunneling, additional headers are added to the traffic that is to be tunneled. Furthermore, various transformations can be applied to the network traffic depending on the type of tunnel interface. When routing traffic over an IPsec interface, for instance, the payload is usually encrypted to achieve confidentiality.

CorePlus supports the following tunnel interface types:

- *IPsec* interfaces are used as end-points for IPsec VPN tunnels. For more information about IPsec VPN, please see Section 9.2.1, “IPsec Basics”.
- *PPTP/L2TP* interfaces are used as end-points for PPTP or L2TP tunnels. For more information about PPTP/L2TP, please see Section 9.4, “PPTP/L2TP”.

- *GRE* interfaces are used to establish GRE tunnels. For more information about GRE, please see Section 3.3.5, “GRE Tunnels”.

### Loopback Interfaces

A *loopback interface* is a special type of interface that will take all packets sent through it and pass them on out through the loopback interface configured as the one to loop to. These are mainly used for *Virtual Routing* scenarios.

For more information about loopback interfaces, please see Section 3.3.6, “Loopback Interfaces”.

Even though the various types of interfaces are very different in the way they are implemented and how they work, CorePlus treats all interfaces as logical IP interfaces. This means that all types of interfaces can be used almost interchangeably in the various subsystems and policies. The result of this is a very high flexibility in how traffic can be controlled and routed in the system.

Each interface in CorePlus is given a unique name to be able to select it into other subsystems. Some of the interface types provide relevant default names that are possible to modify should that be needed, while other interface types require a user-provided name.

### The *any* and *core* interfaces

In addition, CorePlus provides two special logical interfaces named **core** and **any**:

- **any** represents all possible interfaces including the **core** interface
- **core** indicates that it is CorePlus itself that will deal with the traffic. Examples of the use of **core** would be when the Clavister Security Gateway acts as a PPTP or L2TP server or is to respond to ICMP "Ping" requests. By specifying the **Destination Interface** of a route as **core**, CorePlus will then know that it is itself that is the ultimate destination of the traffic.

## 3.3.2. Ethernet

Clavister Security Gateways support Ethernet, Fast Ethernet and Gigabit Ethernet interfaces as defined by the IEEE 802.3 standard.

The IEEE 802.3 Ethernet standard allows various devices to be attached at arbitrary points or 'ports' to a physical transport mechanism such as a coaxial cable. Using the CSMA/CD protocol, each Ethernet connected device 'listens' to the network and sends data to another connected device when no other is sending. If 2 devices broadcast simultaneously, algorithms allow them to re-send at different times. Devices broadcast data as frames and the other devices 'listen' to determine if they are the intended destination for any of these frames.

A frame is a sequence of bits which specify the originating device plus the destination device, the data payload along with error checking bits. A pause between the broadcasting of individual frames allows devices time to process each frame before the next arrives and this pause becomes progressively smaller as the transmission rates get faster from normal to Fast and then Gigabit Ethernet.

Each Ethernet interface in the Clavister Security Gateway corresponds to a physical Ethernet port in the system. The number of ports, their link speed and the way the ports are realized, is of course highly dependent on what type of hardware is used to run the system. A smaller turn-key Clavister device, for instance, is likely to have a limited number of Fast Ethernet ports as integrated product components, while a more powerful unit designed for telecom company usage might be expandable with separate port modules. Furthermore, if CorePlus is being run on off-the-shelf Intel x86 compatible server hardware, it is likely that the Ethernet ports are realized through common Ethernet PCI adapters.

**Note**

*Some systems are using an integrated layer 2 switch for providing additional physical Ethernet ports. Those ports are however seen as a single interface by CorePlus.*

### 3.3.2.1. Ethernet Interface Basics

#### Ethernet Interface Names

The names of the Ethernet interfaces are pre-defined by the system, and are mapped to the names of the physical ports; a system with a *wan* port will have an Ethernet interface named *wan* and so forth.

The names of the Ethernet interfaces can be changed to better reflect their usage. For instance, if an interface named *dmz* is connected to a wireless LAN, it might be convenient to change the interface name to *radio*. For maintenance and troubleshooting, it is recommended to tag the corresponding physical port with the new name.

**Note**

*The startup process will enumerate all available Ethernet interfaces. Each interface will be given a name of the form *lan*, *wan* and *dmz* or *ifN* or *sfpN*, *geN* and *feN* depending on your Clavister Security Gateway model, where *N* represents the number of the interface. In most of the examples in this guide *lan* is used for LAN traffic and *wan* is used for WAN traffic. If your Clavister Security Gateway does not have these interfaces, please substitute the references with the name of your chosen interface.*

#### IP Addresses

Each Ethernet interface is required to have an *Interface IP Address*, which can be either a static address or an address provided by DHCP. The interface IP address is used as the primary address for communicating with the system through the specific Ethernet interface.

The standard is to use IP4 Address objects to define the addresses of Ethernet interfaces. Those objects are normally auto-generated by the system. For more information, please see Section 3.1.5, “Auto-Generated Address Objects”. When the system is first started, all unconfigured Ethernet interfaces will be assigned default addresses from the localhost subnet (i.e. 127.0.x.y).

**Tip**

*Multiple IP addresses can be specified for an Ethernet interface by using the ARP Publish feature. (See section Section 3.4, “ARP”).*

In addition to the interface IP address, a *Broadcast* address is also specified for the Ethernet interface. The Broadcast address is the highest address available on the network. In the case of a 32-address network, the broadcast address is the network address +31, e.g. if the network has an address of 192.168.123.64 / 255.255.255.224, its broadcast address is 192.168.123.95. The broadcast address is the address to which information that is to reach all computers connected to the network is sent.

#### Multicast with Ethernet Interfaces

When setting up an Ethernet interface, the option exists to control the reception of multicast IP packets on that interface. There are three options available for multicast packet handling:

- **Off** - Multicast packets are silently dropped
- **On** - Multicast traffic can always be received by the interface.
- **Auto** - If an IP rule exists in the rule-set which applies to a multicast packet's destination IP address, then the Ethernet interface is automatically enabled to receive multicast packets.

**Auto** is the default behaviour.



### Note

The **all-nets** net object (IP address: 0.0.0.0/0) includes the multicast IP address range (224.0.0.0 => 239.255.255.255). For more information see Section 4.6, “Multicast Routing”.

## Default Gateway

Optionally, a *Default Gateway* address can be specified for an Ethernet interface. This setting tells CorePlus how to reach hosts for which no routes exist. In other words, if a Default Gateway address has been specified, CorePlus will automatically create a default route (destination network 0.0.0.0/0) over the actual interface using the specified gateway. For natural reasons, only one Ethernet interface at a time can be assigned a default gateway.

### 3.3.2.2. Using DHCP on Ethernet Interfaces

CorePlus includes a DHCP client for dynamic assignment of address information. The information that can be set using DHCP includes the IP and broadcast address of the interface, the local network that the interface is attached to, and the default gateway.

All addresses received from the DHCP server are assigned to corresponding IP4Address objects. In this way, dynamically assigned addresses can be used throughout the configuration in the same way as static addresses. By default, the objects in use are the same ones as defined in Section 3.1.5, “Auto-Generated Address Objects”.

#### Example 3.9. Enabling DHCP

##### *Clavister FineTune*

1. Select the **Interfaces > Ethernet** section of the target system in the tree view of the Security Editor.
2. Right-click on the ethernet object of interest and choose **Properties**.
3. Click on the **DHCP** tab.
4. Check the **Enable DHCP Client** control.
5. Click **OK**.

### 3.3.3. Virtual LAN

CorePlus is fully compliant with the IEEE 802.1Q specification for Virtual LANs. On a protocol level, Virtual LANs work by adding a Virtual LAN identifier (VLAN ID) to the Ethernet frame header. The VLAN ID is a number from 0 to 4095 and is used to identify a specific Virtual LAN. In this way, Ethernet frames can belong to different Virtual LANs, but still share the same physical media.

The Virtual LAN support in CorePlus works by defining one or more *Virtual LAN interfaces*. Each Virtual LAN interface is interpreted as a logical interface by the system.

Ethernet frames received by the system are examined for a VLAN ID. If a VLAN ID is found, and a matching Virtual LAN interface has been defined, the system will consider that interface to be the receiving interface for the frame before further processing takes place.

Virtual LANs are useful in several different scenarios, for instance, when filtering is needed between different Virtual LANs in an organization, or when the number of interfaces needs to be expanded. For more information about the latter, please see the section Using Virtual LANs to expand gateway interfaces below.



**Note**

*The number of Virtual LAN interfaces that can be defined in the system is determined by the particular product license you have.*

**Example 3.10. Defining a virtual LAN**

This example defines a virtual LAN using a VLAN ID of 10. Note that this Virtual LAN interface will use the IP address of the corresponding Ethernet interface, as no IP address is specified.

**Clavister FineTune**

1. Select the **Interfaces > Virtual LAN** section of the target system in the tree view of the Security Editor.
2. Choose **New Virtual LAN...** from the context menu, the **Virtual LAN Properties** dialog box will be displayed.
3. Specify a suitable name for the event receiver, for instance VLAN10.
4. Now enter:
  - **Interface:** lan
  - **VLAN:** 10
  - **Broadcast:** all-nets
5. Click **OK**.

## 3.3.4. PPPoE

Point-to-Point Protocol over Ethernet (PPPoE) is a tunneling protocol used for connecting multiple users on an Ethernet network to the Internet through a common serial interface, such as a single DSL line, wireless device or cable modem. All the users on the Ethernet share a common connection, while access control can be done on a per-user basis.

Internet server providers (ISPs) often require customers to connect through PPPoE to their broadband service. Using PPPoE the provider can:

- Implement security and access-control using username/password authentication
- Trace IP addresses to a specific user
- Allocate IP address automatically for PC users (similar to DHCP). IP address provisioning can be per user group

### 3.3.4.1. Overview of PPP

Point-to-Point Protocol (PPP), is a protocol for communication between two computers using a serial interface, such as the case of a personal computer connected through a switched telephone line to an ISP. In terms of the OSI model, PPP provides a layer 2 encapsulation mechanism to allow packets of any protocol to travel through IP networks. PPP uses Link Control Protocol (LCP) for link establishment, configuration and testing. Once the LCP is initialized, one or several Network Control Protocols (NCPs) can be used to transport traffic for a particular protocol suite, so that multiple protocols can interoperate on the same link, for example, both IP and IPX traffic can share a PPP link.

Authentication is an option with PPP. Authentication protocols supported are Password Authentication Protocol (PAP), Challenge Handshake Authentication Protocol (CHAP), Microsoft CHAP (version 1 and 2). If authentication is used, at least one of the peers has to authenticate itself before the network layer protocol parameters can be negotiated using NCP. During the LCP and NCP negotiation, optional parameters such as encryption, can be negotiated.

## 3.3.4.2. PPPoE Client Configuration

### The PPPoE interface

Since the PPPoE protocol runs PPP over Ethernet, the gateway needs to use one of the normal Ethernet interfaces to run PPPoE over. Each PPPoE Tunnel is interpreted as a logical interface by the CorePlus, with the same routing and configuration capabilities as regular interfaces, with the IP rule-set being applied to all traffic. Network traffic arriving at the gateway through the PPPoE tunnel will have the PPPoE tunnel interface as its source interface. For outbound traffic, the PPPoE tunnel interface will be the destination interface. As with any interface, one or more routes are defined so CorePlus knows what IP addresses it should accept traffic from and which to send traffic to through the PPPoE tunnel. The PPPoE client can be configured to use a service name to distinguish between different servers on the same Ethernet network.

### IP address information

PPPoE uses automatic IP address allocation which is similar to DHCP. When CorePlus receives this IP address information from the ISP, it stores it in a network object and uses it as the IP address of the interface.

### User authentication

If user authentication is required by the ISP, the username and password can be setup in CorePlus for automatic sending to the PPPoE server.

### Dial-on-demand

If dial-on-demand is enabled, the PPPoE connection will only be up when there is traffic on the PPPoE interface. It is possible to configure how the gateway should sense activity on the interface, either on outgoing traffic, incoming traffic or both. Also configurable is the time to wait with no activity before the tunnel is disconnected.

#### Example 3.11. Configuring a PPPoE client on the wan interface with traffic routed over PPPoE.

##### *Clavister FineTune*

1. Select the **Interfaces > PPPOEClients** section of the target system in the tree view of the Security Editor.
2. Choose **New PPPoEClient...** from the context menu, the **PPPoEClient Properties** dialog box will be displayed.
3. Specify a suitable name for the object, for instance PPPoEClient.
4. **Physical Interface:** wan
5. **Remote Network:** all-nets (as we will route all traffic into the tunnel)
6. Click on **Authentication** tab and then enter:
  - **Username:** Username provided by the service provider
  - **Password:** Password provided by the service provider
  - **Confirm Password:** Retype the password
  - **Service Name:** Service name provided by the service provider
  - **Authentication** You can specify exactly which authentication protocol to use. The default settings will be used if not specified.

7. Under the **Dial-on-demand** tab, make sure that the **Enable dial-on-demand** control is unchecked.
8. Click the **OK**.

**Note**

*To provide a point-to-point connection over Ethernet, each PPP session must learn the Ethernet address of the remote peer, as well as establish a unique session identifier. PPPoE includes a discovery protocol that provides this.*

## 3.3.5. GRE Tunnels

Generic Router Encapsulation (GRE) protocol is a simple encapsulating protocol with low overhead that provides a method of connecting two networks together which use one network layer protocol, across another network, such as the internet, which is using a different network layer protocol. A GRE tunnel does not use any encryption for the communication and is therefore not, in itself, secure. Each GRE Tunnel is interpreted as a logical interface by CorePlus, with the same filtering, traffic shaping and configuration capabilities as a standard interface.

A typical use of GRE tunneling is where a UDP data stream is to be multicast and it is necessary to transit through a gateway which does not support multicasting.

An established GRE tunnel does not automatically mean that all traffic coming from or to that GRE tunnel is trusted. On the contrary, network traffic coming from the GRE tunnel will be transferred to the CorePlus IP rule-set for evaluation. The source interface of the network traffic will be the name of the associated GRE Tunnel. The same is true traffic in the opposite direction, that is, going into a GRE tunnel. Furthermore a **Route** has to be defined so CorePlus knows what IP addresses should be accepted and sent through the tunnel.

## 3.3.6. Loopback Interfaces

A *Loopback Interface* is an interface that will take all packets sent through it and pass them on out through the Loopback Interface configured as the one to loop to. These are mainly used in CorePlus for *Virtual Routing* scenarios. In Virtual Routing it is possible to divide up any Clavister Security Gateway's operation so that it can behave as multiple virtual gateways. Loopback Interfaces allow the routing of traffic between these virtual gateways.

When creating new Loopback Interfaces, they are always created as a pair, one for each end of the loop.

A Loopback Interface can be configured with IP addresses, just as any other interface type. This IP address is used for communicating with the system and as the source address for dynamically translated connections that are routed over the interface. If the interface will not be used for these purposes, it makes sense to set the IP address to an address in the range of the loopback IP addresses (127.0.0.0 - 127.255.255.255).

### Example 3.12. Creating a Loopback Interface Pair

This example shows how to create a Loopback Interfaces pair.

#### **Clavister FineTune**

1. Select the **Interfaces/Loopback Interfaces** section in the tree view of the Security Editor.
2. Choose **Create Loopback Interface Pair...** from the context menu. The **Create Loopback Interface Pair** dialog box will be displayed.
3. Specify a suitable name for the first loopback interface, for instance **main-vr1**.
4. Specify 127.0.0.1 as IP address for the first Loopback Interface.

5. Specify a suitable name for the second Loopback Interface, for instance **vr1-main**.
6. Specify 127.0.0.2 as the IP address for the second loopback interface.
7. Click **OK**

Two interfaces have now been added; **main-vr1** and **vr1-main**. Packets being sent through the **main-vr1** interface will be received on the **vr1-main** interface and vice versa.

## 3.3.7. Interface Groups

Multiple CorePlus interfaces can be grouped together to form an *Interface Group*. Such a logical group can then be subject to common policies and be referred to using a group name in the IP rule-set and User Authentication Rules.

A group can consist of regular Ethernet interfaces, VLAN interfaces, or VPN Tunnels and the members of a group need not be of the same type. A group might consist, for instance, of two Ethernet interfaces and four VLAN interfaces.

### Example 3.13. Creating an Interface Group

#### *Clavister FineTune*

1. Select the **Interfaces > Interface Group** section of the target system in the tree view of the Security Editor.
2. Choose **New Interface Group...** from the context menu, the **Interface Group Properties** dialog box will be displayed.
3. Specify a suitable name for the interface group, for instance **examplegroup**.
4. Check the **Security/Transport Equivalent:** control. (If enabled, the interface group can be used as a destination interface in rules where connections might need to be moved between the interfaces. Examples of such usage are Route Fail-Over and OSPF).
5. In the **Interfaces** control, use the plus sign and the cross to add or remove interfaces to the group.
6. Click **OK**.

## 3.4. ARP

### 3.4.1. Overview

Address Resolution Protocol (ARP) is a protocol, which maps a network layer protocol address to a data link layer hardware address and it is used to resolve an IP address into its corresponding Ethernet address. It works at the OSI Data Link Layer (Layer 2 - see Appendix F, *The OSI framework*) and is encapsulated by Ethernet headers for transmission.

A host in an Ethernet network can communicate with another host, only if it knows the Ethernet address (MAC address) of that host. A higher level protocols like IP uses IP addresses which are fundamentally different from a lower level hardware addressing scheme like the MAC address. ARP is used to get the Ethernet address of a host from its IP address. When a host needs to resolve an IP address to its Ethernet address, it broadcasts an ARP request packet. The ARP request packet contains the source MAC address and the source IP address and the destination IP address. Each host in the local network receives this packet. The host with the specified destination IP address, sends an ARP reply packet to the originating host with its MAC address.

### 3.4.2. ARP in CorePlus

CorePlus provides not only standard support for ARP, but also adds a number of security checks on top of the protocol implementation. As an example, CorePlus will by default *not* accept ARP replies for which the system has not sent out a corresponding ARP query for. Without this type of protection, the system would be vulnerable to "connection hijacking".

CorePlus supports both dynamic ARP as well as static ARP, and the latter is available in two modes; Publish and XPublish.

*Dynamic ARP* is the main mode of operation for ARP, where CorePlus sends out ARP requests whenever it needs to resolve an IP address to an Ethernet address. The ARP replies are stored in the ARP cache of the system.

*Static ARP* is used for manually lock an IP address to a specific Ethernet address. This is explained in more detail in the sections below.

### 3.4.3. ARP Cache

The *ARP Cache* is the temporary table in CorePlus for storing the mapping between IP and Ethernet addresses. The ARP cache is empty at system startup and will be populated with entries as needed.

The contents of a typical (minimal) ARP Cache looks similar to the following table:

Type	IP Address	Ethernet Address	Expire
Dynamic	192.168.0.10	08:00:10:0f:bc:a5	45
Dynamic	193.13.66.77	0a:46:42:4f:ac:65	136
Publish	10.5.16.3	4a:32:12:6c:89:a4	-

The first item in this ARP Cache is a dynamic ARP entry which tells us that IP address 192.168.0.10 is mapped to an Ethernet address of 08:00:10:0f:bc:a5. The second item dynamically maps the IP address 193.13.66.77 to Ethernet address 0a:46:42:4f:ac:65. Finally, the third item is a static ARP entry binding the IP address 10.5.16.3 to Ethernet address 4a:32:12:6c:89:a4.

The third column in the table, Expire, is used to indicate for how much longer the ARP entry will be valid. The first item, for instance, has an expiry value of 45, which means that this entry will be rendered invalid and removed from the ARP Cache in 45 seconds. If traffic is going to be sent to the 192.168.0.10 IP address after the expiration, CorePlus will issue a new ARP request.

The default expiration time for dynamic ARP entries is 900 seconds (15 minutes). This can be changed by modifying the **ARPExpire** setting in the **Advanced Settings > ARP** section. The set-

ting **ARPExpireUnknown** specifies how long CorePlus is to remember addresses that cannot be reached. This is done to ensure that CorePlus does not continuously request such addresses. The default value for this setting is 3 seconds.

## Displaying the ARP Cache

### Example 3.14. Displaying the ARP Cache

The contents of the ARP Cache can be displayed from within the CLI.

#### CLI

```
Cmd> arp
ARP cache of iface lan
  Dynamic 10.4.0.1      = 1000:0000:4009   Expire=196
  Dynamic 10.4.0.165   = 0002:a529:1f65   Expire=506
```

## Flushing the ARP Cache

If a host in your network has recently been replaced with a new hardware but keeping the same IP address, it is most likely to have a new Ethernet address. If CorePlus has an ARP entry for that host, the Ethernet address of that entry will be invalid, causing data sent to the host to never reach its destination.

Naturally, after the ARP expiration time, CorePlus will learn the new Ethernet address of the requested host, but sometimes it might be necessary to manually force a re-query. This is easiest achieved by *flushing* the ARP cache, an operation which will basically delete all dynamic ARP entries from the cache, thereby forcing CorePlus to issue new ARP queries.

### Example 3.15. Flushing the ARP Cache

This example shows how to flush the ARP Cache from within the CLI.

#### CLI

```
Cmd> arp -flush
ARP cache of all interfaces flushed.
```

## Size of the ARP Cache

By default, the ARP Cache is able to hold 4096 ARP entries at the same time. This is feasible for most deployments, but in rare occasions, such as when there are several very large LANs directly connected to the gateway, it might be necessary to adjust this value. This can be done by by modifying the **ARPCacheSize** setting in the **Advanced Settings > ARP** section.

So-called "hash tables" are used to rapidly look up entries in the ARP Cache. For maximum efficiency, a hash should be twice as large as the table it is indexing, so if the largest directly-connected LAN contains 500 IP addresses, the size of the ARP entry hash should be at least 1000 entries. You can modify the **ARPHashSize** setting in the **Advanced Settings > ARP** section to reflect your network. The default value of the setting is 512.

The **ARPHashSizeVLAN** setting is similar to the **ARPHashSize** setting, but affects the hash size for VLAN interfaces only. The default value is 64.

## 3.4.4. Static and Published ARP Entries

CorePlus supports defining static ARP entries (static binding of IP addresses to Ethernet addresses) as well as publishing IP addresses with a specific Ethernet address.

## Static ARP Entries

Static ARP items may help in situations where a device is reporting incorrect Ethernet address in response to ARP requests. Some workstation bridges, such as radio modems, can have such problems. It may also be used to lock an IP address to a specific Ethernet address for increasing security or to avoid denial-of-service if there are rogue users in a network. Note however, that such protection only applies to packets being sent to that IP address, it does not apply to packets being sent from that IP address.

### Example 3.16. Defining a Static ARP Entry

This example will create a static mapping between IP address `192.168.10.15` and Ethernet address `4b:86:f6:c5:a2:14` on the `lan` interface:

#### *Clavister FineTune*

1. Select the **ARP** section of the target system in the tree view of the Security Editor.
2. Choose **New ARP...** from the context menu, the **ARP Properties** dialog box will be displayed.
3. Select **Static** in the **Mode** dropdown list.
4. Choose **lan** in the **Interface** dropdown list.
5. Enter `192.168.10.15` in the **IP Address** textbox.
6. Enter `4b:86:f6:c5:a2:14` in the **MAC Address** textbox.
7. Click **OK**.

## Published ARP Entries

CorePlus supports *publishing* ARP entries, meaning that you can define IP addresses (and optionally Ethernet addresses) for an interface. CorePlus will then provide ARP replies for ARP requests related to those IP addresses.

This can serve two purposes:

- To give the impression that an interface in CorePlus has more than one IP address.
- To aid nearby network equipment responding to ARP in an incorrect manner. This use is however less common.

The first purpose is useful if there are several separate IP spans on a single LAN. The hosts on each IP span may then use a gateway in their own span when these gateway addresses are published on the corresponding CorePlus interface.

Another use is publishing multiple addresses on an external interface, enabling CorePlus to statically address translate communications to these addresses and send it onwards to internal servers with private IP addresses.

There are two publishing modes; Publish and XPublish. The difference between the two is that XPublish "lies" about the sender Ethernet address in the Ethernet header; this is set to be the same as the published Ethernet address rather than the actual Ethernet address of the Ethernet interface. If a published Ethernet address is the same as the Ethernet address of the interface, it will make no difference if you select Publish or XPublish, the result will be the same.



#### **Tip**

*In the configuration of ARP entries, addresses may only be published one at a time. However, you can use the Section 4.2.3, "Proxy ARP" feature to handle publishing of entire networks.*

## 3.4.5. Advanced ARP Settings

This section presents some of the more advanced settings related to ARP. In most cases, these settings need not to be changed, but in some specific deployments, modifications might be needed.

### Multicast and Broadcast

ARP requests and ARP replies containing multicast or broadcast addresses are usually never correct, with the exception of certain load balancing and redundancy devices, which make use of hardware layer multicast addresses.

The default behavior of CorePlus is to drop and log such ARP requests and ARP replies. This can however be changed by modifying the **ARPMulticast** and **ARPBroadcast** settings in the **Advanced Settings > ARP** section.

### Unsolicited ARP Replies

It is fully possible for a host on the LAN to send an ARP reply to the gateway, even though a corresponding ARP request has not been issued. According to the ARP specification, the recipient should accept these types of ARP replies. However, because this can facilitate hijacking of local connections, CorePlus will normally drop and log such replies.

The behavior can be changed by modifying the **UnsolicitedARPReplies** setting in the **Advanced Settings > ARP** section.

### ARP Requests

The ARP specification states that a host should update its ARP Cache with data from ARP requests received from other hosts. However, as this procedure can facilitate hijacking of local connections, CorePlus will normally not allow this.

To make the behavior compliant with the RFC 826 specification, you can modify the **ARPRequests** setting in the **Advanced Settings > ARP** section. Even if ARPRequests is set to "Drop", meaning that the packet is discarded without being stored, the system will, provided that other rules approve the request, reply to it.

### Changes to the ARP Cache

CorePlus provides a few settings controlling how to manage changes to the ARP cache.

Possibly, a received ARP reply or ARP request would alter an existing item in the ARP cache. Allowing this to take place may facilitate hijacking of local connections. However, not allowing this may cause problems if, for example, a network adapter is replaced, as CorePlus will not accept the new address until the previous ARP cache entry has timed out.

The **ARPChanges** setting in the **Advanced Settings > ARP** section can be adjusted to change the behavior. The default is that CorePlus will allow changes to take place, but all such changes will be logged.

Another, similar, situation is where information in ARP replies or ARP requests would collide with static entries in the ARP cache. Naturally, this is never allowed to happen. However, the setting **StaticARPChanges** in the **Advanced Settings > ARP** section does allow you to specify whether or not such situations are to be logged.

### Sender IP 0.0.0.0

CorePlus can be configured on what to do with ARP queries that have a sender IP of 0.0.0.0. Such sender IPs are never valid in responses, but network units that have not yet learned of their IP address sometimes ask ARP questions with an "unspecified" sender IP. Normally, these ARP replies are dropped and logged, but the behavior can be changed by modifying the **ARPQueryNoSenderIP** setting in the **Advanced Settings > ARP** section.



## Matching Ethernet Addresses

By default, CorePlus will require that the sender address at Ethernet level should comply with the Ethernet address reported in the ARP data. If this is not the case, the reply will be dropped and logged. Change the behavior by modifying the **ARPMatchEnetSender** setting in the **Advanced Settings > ARP** section.

## 3.5. The IP Rule-Set

### 3.5.1. Overview

Security policies designed by the administrator regulate the way in which network applications are protected against abuse and inappropriate use. CorePlus provides an array of mechanisms and logical constructs to help with the building of such policies for attack prevention, privacy protection, identification, and access control.

IP Rules are at the heart of creating a security policy. Once logical constructs such as Application Layer Gateways are created, they don't have any effect on traffic flow until they are used somewhere in the IP rule-set. Understanding how to define IP rules is therefore crucial to understanding how to create overall security policies.

The list of rules defined on the basis of network objects such as addresses, protocols and services, form the central function of the Clavister Security Gateway. IP Rules determine the basic and essential filtering functions of the product. By following the rules configuration, CorePlus regulates what is allowed or not allowed to pass through the Clavister Security Gateway, and how address translation, bandwidth management and traffic shaping are applied to the traffic flow. It is important for the administrator to understand how IP Rules function since ambiguous or faulty rules can lead to breaches in the secure operation of the gateway.



#### **Note**

*The list of IP rules defined by the administrator is also referred to in Clavister documentation simply as the "rule-set".*

There are two essential stances which describe the underlying philosophy of security in the Clavister Security Gateway:

- Everything is denied unless specifically permitted
- Everything is permitted unless specifically denied

In order to provide the highest possible level of security, **Deny** is the default policy in the Clavister rule-set. The default of deny is accomplished without a visible rule in the list. However, for logging purposes, the rule list usually has a **DropAll** rule at the bottom with logging enabled.

### 3.5.2. Rule Evaluation

When a new TCP/IP connection is being established through the Clavister Security Gateway, the list of rules are evaluated from top to bottom until a rule that matches the parameters of that new connection is found. Those parameters include, amongst others, the source IP address and the destination IP address plus the source gateway interface and the destination gateway interface. The **Action** of the rule is then carried out by CorePlus.

If the rule action is **Allow** then the establishment of the new connection will be permitted. Furthermore, an entry representing that new connection is added to the gateway's internal "state table" which allows monitoring of opened and active connections passing through the gateway. If, instead, the action were **Drop**, the new connection will be refused.

### The First Matching Principle

If several rules match the connections parameters, the first matching rule in the scan from top to bottom of the list, is the rule that decides what will happen with the connection. (The exception being SAT rules.)

After initial rule evaluation of the opening connection, subsequent packets belonging to that connec-

tion will not need to be evaluated again against the rule-set. Instead, a highly efficient algorithm searches the internal state table for an existing state representing the connection to which the packets belongs to determine if it is part of an already established link that has been already passed through the rule-set. This approach is known as "stateful inspection" and is applied not only to stateful protocols such as TCP connections, but also, by means of "pseudo-connections" to stateless protocols such as UDP and ICMP as well.

The state-inspection approach in Clavister Security Gateways means that evaluation against the rule-set is only done in the initial opening phase of a connection. The size of the rule-set consequently has almost no effect on the overall throughput performance of the gateway.

### 3.5.3. Rule components

A rule consists of two logical parts: the connection parameters and the action to take if there is a match with those parameters.

Rule parameters are pre-defined and reusable network objects such as **Addresses** and **Services**, which can be used in any rule to specify the criteria for a match.

#### Rule Parameters

The following parameters are set for a single rule. There has to be a match with all parameters in a rule for that rule to be triggered.

<b>Service</b>	The protocol type to which the packet belongs. Services are defined as logical objects before configuring the rules.
<b>Source Interface</b>	An <b>Interface</b> or <b>Interface Group</b> where the packet is received on the gateway.
<b>Source Network</b>	The network that contains the source IP address of the packet.
<b>Destination Interface</b>	An <b>Interface</b> or an <b>Interface Group</b> from which the packet would leave the gateway.
<b>Destination Network</b>	The network to which the destination IP address of the packet belongs.

#### Actions

When a rule is triggered by a match with it's parameters any one of the following can occur:

<b>Allow</b>	The packet is allowed to pass. As the rule is applied to only the opening of a connection, an entry in the "state table" is made to record that a connection is open. The remaining packets related to this connection will pass through the gateway's "stateful inspection engine".
<b>NAT</b>	This functions like an <b>Allow</b> rule, but with dynamic address translation (NAT) enabled. See Section 7.1, "Dynamic Address Translation (NAT)".
<b>FwdFast</b>	Let the packet pass through the gateway without setting up a state for it. This means that the stateful inspection process is bypassed and is therefore less secure than <b>Allow</b> or <b>NAT</b> rules. Packet processing is also slower than <b>Allow</b> rules as every packet is checked against the entire rule-set.
<b>SAT</b>	Tells CorePlus to perform static address translation. A <b>SAT</b> rule also requires a matching <b>Allow</b> , <b>NAT</b> or <b>FwdFast</b> rule further down the rule-set. See Section 7.2, "Static Address Translation (SAT)".

**Drop** Tells CorePlus to immediately discard the packet.

**Reject** Acts like **Drop**, but will return a "TCP RST" or "ICMP Unreachable message", informing the sending computer that the packet was disallowed.



***Note***

*Packets not matching a rule in the rule-set and not having an already opened matching connection in the "state table" will be dropped.*

## 3.6. Schedules

In some scenarios, it might be useful to control not only what functionality is enabled, but also when that functionality is being used.

For instance, the IT policy of an enterprise might stipulate that web traffic from a certain department is only allowed access outside that department during normal office hours. Another example might be that authentication using a specific VPN connection is only permitted on weekdays before noon.

CorePlus addresses this requirement by providing *Schedule* objects, or simply *schedules*, that can be selected into various types of policies to accomplish time-based control. This functionality is in no way limited to IP Rules, but is valid for most types of policies, including Traffic Shaping rules, Intrusion Detection and prevention (IDP) rules and Virtual Routing rules. A Schedule object is, in other words, a very powerful component that can allow detailed regulation of when functions in CorePlus are enabled or disabled.

A Schedule object gives the possibility to enter multiple time ranges for each day of the week. Furthermore, a start and a stop date can be specified that will impose additional constraints on the schedule. For instance, a schedule can be defined as Mondays and Tuesdays, 08:30 - 10:40 and 11:30 - 14:00, Fridays 14:30 - 17:00.



### **Important**

*As schedules depend on an accurate date and time, it's very important that the system date and time are set correctly. Preferably, time synchronization has also been enabled to ensure that scheduled policies will be enabled and disabled at the right time. For more information, please see Section 3.8.1, "Date and Time".*

### **Example 3.17. Setting up a Time-Scheduled Policy**

This example creates a schedule object for office hours on weekdays, and attaches the object to an IP Rule that allows HTTP traffic.

#### **Clavister FineTune**

1. Select the **Schedule Profiles** section of the target system.
2. Choose **New Schedule Profile...** from the context menu, the **Schedule Profile Properties** dialog box will be displayed.
3. Specify a suitable name for the schedule profile, for instance OfficeHours.
4. Select 08-17, Monday to Friday in the grid.
5. Click **OK**.

Now attach the object to an IP rule:

1. Select the **Rules** section of the target system.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the schedule profile, for instance OfficeHours.
4. Select the following from the dropdown lists:
  - **Action:** NAT
  - **Service:** http
  - **Schedule:** OfficeHours
  - **SourceInterface:** lan
  - **SourceNetwork:** lannet

- **DestinationInterface:** any
  - **DestinationNetwork:** all-nets
5. Click **OK**.

## 3.7. X.509 Certificates

CorePlus supports digital certificates that comply with the ITU-T X.509 standard. This involves the use of an X.509 certificate hierarchy with public-key cryptography to accomplish key distribution and entity authentication.

### 3.7.1. Overview

An X.509 certificate is a digital proof of identity. It links an identity to a public key in order to establish whether a public key truly belongs to the supposed owner. By doing this, it prevents data transfer interception by a malicious third-party who might post a phony key with the name and user ID of an intended recipient.

A certificate consists of the following:

- A public key: The "identity" of the user, such as name, user ID.
- Digital signatures: A statement that tells the information enclosed in the certificate has been vouched for by a Certificate Authority (CA).

By binding the above information together, a certificate is a public key with identification attached, coupled with a stamp of approval by a trusted party.

### 3.7.2. The Certification Authority

A certification authority ("CA") is a trusted entity that issues certificates to other entities. The CA digitally signs all certificates it issues. A valid CA signature in a certificate verifies the identity of the certificate holder, and guarantees that the certificate has not been tampered with by any third party.

A certification authority is responsible for making sure that the information in every certificate it issues is correct. It also has to make sure that the identity of the certificate matches the identity of the certificate holder.

A CA can also issue certificates to other CAs. This leads to a tree-like certificate hierarchy. The highest CA is called the root CA. In this hierarchy, each CA is signed by the CA directly above it, except for the root CA, which is typically signed by itself.

A certification path refers to the path of certificates from one certificate to another. When verifying the validity of a user certificate, the entire path from the user certificate up to the trusted root certificate has to be examined before establishing the validity of the user certificate.

The CA certificate is just like any other certificates, except that it allows the corresponding private key to sign other certificates. Should the private key of the CA be compromised, the whole CA, including every certificate it has signed, is also compromised.

### 3.7.3. Validity Time

A certificate is not valid forever. Each certificate contains the dates between which the certificate is valid. When this validity period expires, the certificate can no longer be used, and a new certificate has to be issued.

### 3.7.4. Certificate Revocation Lists

A certificate revocation list ("CRL") contains a list of all certificates that have been cancelled before their expiration date. This can happen for several reasons. One reason could be that the keys of the certificate have been compromised in some way, or perhaps that the owner of the certificate has lost

the rights to authenticate using that certificate. This could happen, for instance, if an employee has left the company from whom the certificate was issued.

A CRL is regularly published on a server that all certificate users can access, using either the LDAP or HTTP protocols.

Certificates often contain a CRL Distribution Point (CDP) field, which specifies the location from where the CRL can be downloaded. In some cases certificates do not contain this field. In those cases the location of the CRL has to be configured manually.

The CA updates its CRL at a given interval. The length of this interval depends on how the CA is configured. Typically, this is somewhere between an hour to several days.

## 3.7.5. Trusting Certificates

When using certificates, the gateway trusts anyone whose certificate is signed by a given CA. Before a certificate is accepted, the following steps are taken to verify the validity of the certificate:

- Construct a certification path up to the trusted root CA.
- Verify the signatures of all certificates in the certification path.
- Fetch the CRL for each certificate to verify that none of the certificates have been revoked.

## 3.7.6. Identification Lists

In addition to verifying the signatures of certificates, CorePlus also employs identification lists. An identification list is a list naming all the remote identities that are allowed access through a specific VPN tunnel, provided the certificate validation procedure described above succeeded.

## 3.7.7. X.509 Certificates in CorePlus

X.509 certificates can be uploaded to the Clavister Security Gateway for use in IKE/IPsec authentication, webauth etc. There are two types of certificates that can be uploaded, self signed certificates and remote certificates belonging to a remote peer or CA server.



## 3.8. System Settings

### 3.8.1. Date and Time

Correct setting of the date and time is important for the product to operate properly. For instance, time scheduled policies and auto-update of IDP signatures are two features that require the clock to be accurate. In addition, log messages are tagged with time-stamps in order to indicate when a specific event occurred. Not only does this assume a working clock, but also that the clock is being synchronized with other devices in the network.

To maintain current date and time, the product makes use of a built-in real-time clock. The clock is also equipped with a backup battery to ensure operation even if there's a temporary loss of power. In addition, the product supports time synchronization protocols in order to automatically adjust the clock, based on information from other devices.

#### 3.8.1.1. General Date and Time Settings

##### Current Date and Time

###### Example 3.18. Setting the Current Date and Time

To adjust the current date and time, follow the steps outlined below:

###### CLI

```
Cmd> time -set YYYY-mm-DD HH:MM:SS
```

Where YYYY-mm-DD HH:MM:SS is the new date and time.



###### Note

A new date and time will be applied as soon as it's set.

##### Time Zone

The time zone setting should be set to reflect the time zone where the product is physically located.

###### Example 3.19. Setting the Time Zone

To modify the time zone, follow the steps outlined below:

###### Clavister FineTune

1. Select the **Advanced Settings/Timesync** section in the tree view of the Security Editor.
2. In the **TimeSync\_TimeZoneOffs** setting, enter the amount of minutes corresponding to the actual timezone. For instance, if the timezone is GMT+1 (GMT plus 60 minutes), the value of the setting should be 60.

##### Daylight Saving Time

Many regions honor Daylight Saving Time (DST) (or summer time as it is called in many countries). Daylight saving time works by advancing the clock during summer to get more out of the summer days. Unfortunately, the principles regulating daylight saving time vary in different countries, and in some cases there are even variants within the same country. For this reason, the product does not automatically know when to adjust for DST. Instead, this information has to be manually provided if daylight saving time is to be used.

There are basically two parameters governing daylight saving time; the DST period and the DST offset. The DST period specifies on what dates daylight saving time starts and ends, respectively. The DST offset indicates the number of minutes to advance the clock during the daylight saving time period.

#### Example 3.20. Enabling DST

To enable DST, follow the steps outlined below:

##### *Clavister FineTune*

1. Select the **Advanced Settings/Timesync** section in the tree view of the Security Editor.
2. Check the **TimeSync\_DSTEnabled** checkbox.
3. In the **TimeSync\_DSTOffs** setting, enter the DST offset in minutes.
4. In the **TimeSync\_DSTStartDate** and **TimeSync\_DSTEndDate** settings, enter the dates that specify when the DST period starts and ends, respectively.

### 3.8.1.2. Time Synchronization

The clock in the product is likely to be fast or slow after a period of operation. This is normal behavior in most network and computer equipment and is commonly solved by utilizing a time synchronization mechanism.

The product is able to adjust the clock automatically based on information received from one or several timeservers in the network. Using time synchronization is highly recommended, as it ensures the product to have its date and time aligned with other products in the network, or even with public timeservers providing highly accurate time information based on atomic clocks.

#### Time Synchronization Protocols

The product supports two kinds of protocols to be used for time synchronization:

- **SNTP** - Defined by RFC 2030, The Simple Network Time Protocol (SNTP) is a lightweight implementation of the Network Time Protocol (NTP) described in RFC 1305.
- **UDP/TIME** - The Time Protocol (UDP/TIME) is an older method of providing time synchronization service over the Internet. The protocol provides a site-independent, machine-readable date and time. The time service sends back to the originating source the time in seconds since midnight on January first 1900.

Most current public timeservers are using the NTP protocol.

#### Time Servers

Up to three timeservers can be configured to query for time information. By using more than one single server, situations where an unreachable server causes the time synchronization process to fail can be prevented. Please note that the product always queries all configured timeservers in order to compute an average time based on the responses from all servers. Search engines on the Internet can be used to find updated lists of publicly available timeservers.

#### Example 3.21. Enabling Time Synchronization using SNTP

In this example, time synchronization is being setup using the SNTP protocol and using NTP servers installed at the Swedish national laboratory for time and frequency.

##### *Clavister FineTune*

1. Select the **Advanced Settings/Timesync** section in the tree view of the Security Editor.
2. In the **TimeSync\_ServerType** setting, select SNTP in the dropdown list.
3. In the **TimeSync\_TimeServerIP1** setting, enter ntp1.sp.se.
4. In the **TimeSync\_TimeServerIP2** setting, enter ntp2.sp.se.



### Caution

The above example uses domain names instead of IP addresses. Therefore, make sure the DNS client settings of the system are properly configured as described in Section 3.8.2, “DNS”.

### Example 3.22. Manually Triggering a Time Synchronization

Time synchronization can be triggered from the CLI:

#### CLI

```
Cmd> time -sync
```

## Maximum Adjustment

To avoid situations where a faulty timeserver causes the product to update its clock with highly erroneous time data, a maximum adjustment value (in seconds) can be specified. If the difference between the current time in the product and the time received from a timeserver is greater than the maximum adjustment value, that timeserver response will be discarded. For example, assume that the maximum adjustment value is set to 60 seconds, and that the current time in the product is 16:42:35. If a timeserver responds with a time of 16:43:38, the difference is 63 seconds, which is not acceptable according to the maximum adjustment. Thus, no update will occur for that response.

### Example 3.23. Modifying the Maximum Adjustment Value

#### Clavister FineTune

1. Select the **Advanced Settings/Timesync** section in the tree view of the Security Editor.
2. In the **TimeSync\_MaxAdjust** setting, enter the maximum time drift that a server is allowed to adjust in seconds.

Sometimes it might be necessary to override the maximum adjustment, for instance, if time synchronization has just been enabled and the initial time difference is greater than the maximum adjustment value. It is then possible to manually force a synchronization and disregard the maximum adjustment parameter.

### Example 3.24. Forcing Time Synchronization

This example demonstrates how to force time synchronization, without respecting the maximum adjustment setting.

#### CLI

```
Cmd> time -sync -force
```

## Synchronization Interval

The interval between each synchronization attempt can be adjusted if needed. By default, this value is 86,400 seconds (1 day), meaning that the time synchronization process is executed once per day.

## 3.8.2. DNS

DNS names can be used in various areas of the configuration where IP addresses are unknown, or where it simply makes more sense to make use of DNS resolution instead of static addresses.

To accomplish DNS resolving, CorePlus has a built-in DNS client that can be configured to make use of up to three DNS servers.

### Example 3.25. Configuring DNS Servers

In this example, the DNS client is configured to use one primary and one secondary DNS server, having IP addresses 10.0.0.1 and 10.0.0.2 respectively.

#### *Clavister FineTune*

1. Select the **Advanced Settings/DNSClient** section in the tree view of the Security Editor.
2. In the **DNS\_DNSServerIP1** setting, enter 10.0.0.1.
3. In the **DNS\_DNSServerIP2** setting, enter 10.0.0.2.



---

# Chapter 4. Routing

This chapter describes how to configure IP routing in CorePlus.

- Overview, page 65
- Static Routing, page 66
- Policy-based Routing, page 73
- Virtual Routing, page 78
- Dynamic Routing, page 84
- Multicast Routing, page 91
- Transparent Mode, page 99

## 4.1. Overview

IP routing capabilities belong to the most fundamental functionalities of CorePlus: any IP packet flowing through the system will be subjected to at least one routing decision at some point in time, and proper setup of routing is crucial for a CorePlus system to function as expected.

Apart from basic *Static Routing*, CorePlus also supports *Virtual Routing* and *Dynamic Routing*. In addition, routes can be actively monitored to achieve route and link redundancy with fail-over capabilities.

## 4.2. Static Routing

The most basic form of routing is known as *Static Routing*. The term static refers to the fact that entries in the routing table are manually added and are therefore permanent (or static) by nature.

Due to this manual approach, static routing is most appropriate to use in smaller network deployments where addresses are fairly fixed and where the amount of connected networks are limited to a few. For larger networks however (or whenever the network topology is complex), the work of manually maintaining static routing tables will be time-consuming and problematic. As a consequence, dynamic routing should be used in those cases.

For more information about the dynamic routing capabilities of CorePlus, please see Section 4.5, “Dynamic Routing”. Note however, that even if you choose to implement dynamic routing for your network, you will still need to understand the principles of static routing and how it is implemented in CorePlus.

### The Principles of Routing

IP routing is essentially the mechanism in TCP/IP networks used for delivering IP packets from their source to their ultimate destination through a number of intermediary nodes, most often referred to as routers or gateways. In each router, a *routing table* is consulted to find out where to send the packet next. A routing table usually consists of several *routes*, where each route in principle contains a destination network, an interface to forward the packet on and optionally the IP address of the next gateway in the path to the destination.

The images below illustrates a typical Clavister Security Gateway deployment and how the associated routing table would look like.

Route #	Interface	Destination	Gateway
1	lan	192.168.0.0/24	
2	dmz	10.4.0.0/16	
3	wan	195.66.77.0/24	
4	wan	0.0.0.0/0	195.66.77.4

Basically, this routing table provides the following information:

- Route #1: All packets going to hosts on the 192.168.0.0/24 network should be sent out on the lan interface. As no gateway is specified for the route entry, the host is assumed to be located on the network segment directly reachable from the lan interface.
- Route #2: All packets going to hosts on the 10.4.0.0/16 network are to be sent out on the dmz interface. Also for this route, no gateway is specified.
- Route #3: All packets going to hosts on the 195.66.77.0/24 network will be sent out on the wan interface. No gateway is required to reach the hosts.
- Route #4: All packets going to any host (the 0.0.0.0/0 network will match all hosts) will be sent out on the wan interface and to the gateway with IP address 195.66.77.4. That gateway will then consult its routing table to find out where to send the packets next. A route with destination 0.0.0.0/0 is often referred to as the *Default Route* as it will match all packets for which no specific route has been configured.

When a routing table is evaluated, the ordering of the routes is important. In general, a routing table is evaluated with the most *specific* routes first. In other words, if two routes have destination networks that overlap, the more narrow network will be evaluated prior to the wider one. In the above example, a packet with a destination IP address of 192.168.0.4 will theoretically match both the first route and the last one. However, the first route entry is a more specific match, so the evaluation will end there and the packet will be routed according to that entry.

## 4.2.1. Static Routing in CorePlus

This section describes how routing is implemented in CorePlus, and how to configure static routing.

CorePlus supports multiple routing tables. One main table called **main** is defined by default and is always present. However, more and completely separate routing tables can be defined by the user.

Extra user-defined routing tables can be used in two ways:

- **Virtual Routing** means that logical interfaces can be associated with a particular routing table. This enables a single physical Clavister Security Gateway to act as multiple virtual gateways. Communication between these virtual gateways is achieved through *Loopback Interfaces* (see Section 3.3.6, “Loopback Interfaces”).
- **Policy Based Routing** means that rules in the IP rule-set can be defined which decide which of the routing tables will deal with certain types of traffic (see Section 4.3, “Policy-based Routing”).

### Route Lookup Mechanism

The route lookup mechanism in CorePlus is a bit different compared to how ordinary router products work. In a common router, where the IP packets are forwarded without context (in other words, the forwarding is stateless), the routing table is evaluated for each and every IP packet received by the router. In CorePlus, however, packets are forwarded with state-awareness, so the route lookup process is tightly integrated into the stateful packet mechanisms.

When an IP packet is received on any of the interfaces, the connection table is consulted to see if there is already an open connection for which the received packet belongs. If an existing connection is found, the connection data includes information on where to route the packet. In other words, for already established connections, there is no need for lookups in the routing table. This way of handling routing is far more efficient than traditional routing table lookups, and is one reason for the high forwarding performance of a CorePlus system.

If an established connection cannot be found, then the routing table is consulted. It is important to understand that the route lookup is actually performed *before* the various rules sections get evaluated. As a result of this, the destination interface is known at the time CorePlus is about to decide if the connection should be allowed or dropped. This design allows for more a fine-granular control over filters and policies.

### Route Notation

CorePlus uses a slightly different way of describing routes compared to most other systems. However, we believe that this way of describing routes is easier to understand, making it less likely for users to cause errors or breaches in security.

Most systems do not use the specific interface in the routing table, but specifies the IP address of the interface instead. The below routing table is from a Microsoft Windows XP workstation:

```

=====
Interface List
0x1 ..... MS TCP Loopback interface
0x10003 ...00 13 d4 51 8d dd ..... Intel(R) PRO/1000 CT Network
0x20004 ...00 53 45 00 00 00 ..... WAN (PPP/SLIP) Interface
=====
Active Routes:
Network Destination          Netmask          Gateway          Interface        Metric
-----
0.0.0.0                    0.0.0.0          192.168.0.1      192.168.0.10     20
10.0.0.0                   255.0.0.0         10.4.2.143       10.4.2.143       1
10.4.2.143                 255.255.255.255  127.0.0.1        127.0.0.1        50
10.255.255.255             255.255.255.255  10.4.2.143       10.4.2.143       50
85.11.194.33              255.255.255.255  192.168.0.1      192.168.0.10     20
=====

```



```

      127.0.0.0          255.0.0.0      127.0.0.1      127.0.0.1      1
      192.168.0.0      255.255.255.0 192.168.0.10   192.168.0.10   20
      192.168.0.10     255.255.255.255 127.0.0.1      127.0.0.1      20
      192.168.0.255    255.255.255.255 192.168.0.10   192.168.0.10   20
      224.0.0.0        240.0.0.0      10.4.2.143     10.4.2.143     50
      224.0.0.0        240.0.0.0      192.168.0.10   192.168.0.10   20
      255.255.255.255  255.255.255.255 10.4.2.143     10.4.2.143     1
      255.255.255.255  255.255.255.255 192.168.0.10   192.168.0.10   1
Default Gateway:      192.168.0.1
=====
Persistent Routes:
None

```

The corresponding routing table in CorePlus would have looked similar to this:

Flags	Network	Iface	Gateway	Local IP	Metric
	192.168.0.0/24	lan			20
	10.0.0.0/8	wan			1
	0.0.0.0/0	wan	192.168.0.1		20

Obviously, the CorePlus way of describing the routes is far more easy to read and understand. Apart from this, another advantage with this form of notation is that you can specify a gateway for a particular route, without having a route that covers the gateway's IP address or despite the fact that the route covers the gateway's IP address is normally routed via another interface.

It is also worth mentioning that CorePlus allows you to specify routes for destinations that are not aligned with traditional subnet masks. In other words, it is perfectly legal to specify one route for the destination address range 192.168.0.5-192.168.0.17 and another route for addresses 192.168.0.18-192.168.0.254. This is a feature that makes CorePlus highly suitable for routing tasks in very complex network topologies.

## Displaying the Routing Table

It is important to distinguish between the routing table that is active in the system, and the routing table that you configure. The routing table that you configure contains only the routes that you have added manually (in other words, the static routes). The content of the active routing table, however, will vary depending on several factors. For instance, if dynamic routing has been enabled, the routing table will be populated with routes learned by communicating with other routers in the network. Also, features such as route fail-over will cause the active routing table to look different from time to time.

### Example 4.1. Displaying the Routing Table

This example illustrates how to display the contents of the configured routing table as well as the active routing table.

#### *Clavister FineTune*

The configured routing table can be seen in the Clavister FineTune:

1. Go to **Routing > Routes**
2. A list the currently configured routes will be displayed

#### *CLI*

To see the active routing table enter:

```
Cmd> routes
```

Flags	Network	Iface	Gateway	Local IP	Metric
	192.168.0.0/24	lan			0
	213.124.165.0/24	wan			0
	0.0.0.0/0	wan	213.124.165.1		0

## Initial Static Routes

When the Clavister Security Gateway is configured for the first time, the routing table will initially be empty and it is necessary to enter at least one route for traffic to flow through the gateway

On initial setup of CorePlus through the console, it is however possible to enter an initial *Default Gateway* and this is usually done if the management workstation is at least one router hop away from the gateway. (If this is the case then the *Allowed Management Net* should also be entered via the console). This will automatically create the corresponding route in the routing table.

## Core Routes

CorePlus automatically populates the active routing table with *Core Routes*. These routes are present for the system to understand where to route traffic that is destined for the system itself. There is one route added for each interface in the system. In other words, two interfaces named lan and wan, and with IP addresses 192.168.0.10 and 193.55.66.77, respectively, will result in the following routes:

Route #	Interface	Destination	Gateway
1	core	192.168.0.10	
2	core	193.55.66.77	

Basically, when the system receives an IP packet whose destination address is one of the interface IPs, the packet will be routed to the core interface, i.e. intercepted by the system itself.

There is also a core route added for all multicast addresses:

Route #	Interface	Destination	Gateway
1	core	224.0.0.0/4	

To include the core routes when you display the active routing table, you have to specify an option to the routing command.

### Example 4.2. Displaying the Core Routes

This example illustrates how to display the core routes in the active routing table.

#### CLI

```
Cmd> routes -all
```

Flags	Network	Iface	Gateway	Local IP	Metric
	127.0.0.1	core	(Shared IP)		0
	192.168.0.1	core	(Iface IP)		0
	213.124.165.181	core	(Iface IP)		0
	127.0.3.1	core	(Iface IP)		0
	127.0.4.1	core	(Iface IP)		0
	192.168.0.0/24	lan			0
	213.124.165.0/24	wan			0
	224.0.0.0/4	core	(Iface IP)		0
	0.0.0.0/0	wan	213.124.165.1		0

**Tip**

For detailed information about the output of the CLI `routes` command, please see the CLI Reference Guide.

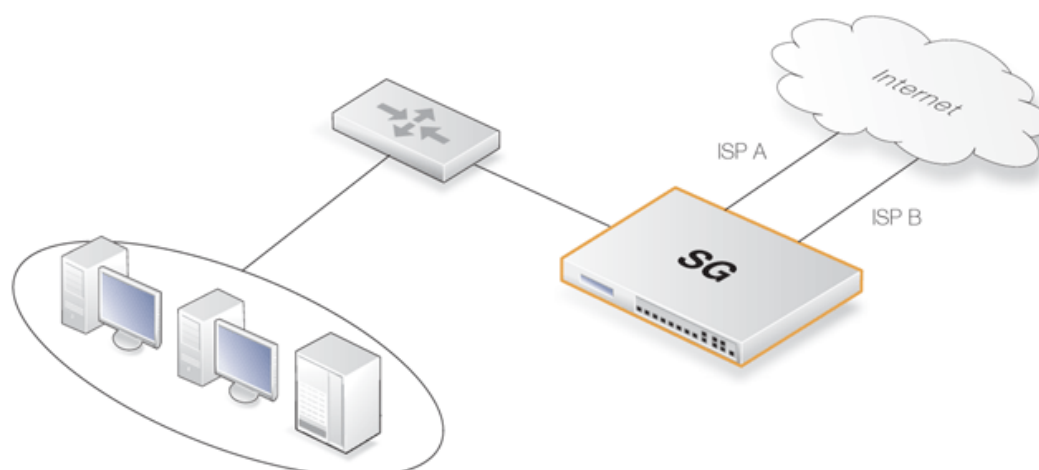
## 4.2.2. Route Failover

Clavister Security Gateways are often deployed in mission-critical locations where availability and connectivity is crucial. A corporation relying heavily on access to the Internet, for instance, could have their operations severely damaged if the Internet connection is down for whatever reason.

As a consequence, it is quite common to find deployments with backup connectivity using a secondary Internet Service Provider. It is also likely that the two service providers use different access methods to avoid single point of failures.

To deal with scenarios such as this, CorePlus provides *Route Failover* so that should connection fail then traffic can *failover* to another, alternate route. CorePlus implements Route Failover through the use of *Route Monitoring*. Through Route Monitoring CorePlus can monitor the health of routes and pick an alternate route should the preferred one be not available.

**Figure 4.1. A Route Failover Scenario for ISP Access**



To activate the Route Failover feature, Route monitoring should be enabled on a per-route basis. For each route there are two monitoring methods available and at least one has to be chosen for monitoring to function:

### Interface Link Status

The system will monitor the link status of the interface specified in the route. As long as the interface is up, the route is diagnosed as healthy. This method is appropriate if you wish to monitor that the interface is physically attached and that the cabling is working as expected. As any changes to the link status are instantly noticed, this method provides the fastest response to failure.

### Gateway Monitoring

If a gateway has been specified for the route, that gateway can be monitored by sending periodic ARP requests. As long as the gateway responds to the requests, the route is considered to be working.

Whenever a monitor diagnoses a route have failed, CorePlus will mark the route as disabled. For new connections, the route lookup will ignore any disabled route and instead find the second best matching route. For already established connections, a route lookup will be performed and the connections will be updated so they start using the new route and Route Failover is achieved.

The table below defines two default routes, both having 0.0.0.0/0 as destination, but using two different gateways. In addition, the first route has a lower metric specified. Route monitoring has been enabled for the first, primary route. Monitoring for the second alternate route isn't meaningful since it has no failover route. If there were a failover route (3 routes in total with two backup routes) then monitoring would be on for the second route.

Route #	Interface	Destination	Gateway	Metric	Monitoring
1	wan	0.0.0.0/0	195.66.77.34	10	On
2	wan	0.0.0.0/0	193.54.68.129	20	Off

When a new connection is about to be established to a host on the Internet, a route lookup will result in that the first route that route has the lowest metric being chosen. Then, if the primary WAN router should fail, this will be detected by the route monitor and the first route will be disabled. As a consequence, a new route lookup will be performed. This time, the second route will be selected as the first one has been marked as disabled by CorePlus

Even if a route has been disabled, the route monitor will continue to check the status of the route. Should the route be diagnosed as healthy again, it will be re-enabled and connections will be transferred back to that route.

## Routing Monitoring and Interface Groups

When using route monitoring, you have to pay attention to if a failover to another route will cause the routing interface to be changed. If so, you will need to take some preventive steps to ensure that policies and existing connections will be maintained.

To illustrate the problem, consider the following configuration:

First, you have one IP rule that will NAT all HTTP traffic destined for the Internet through the wan interface:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	NAT	lan	lannet	wan	all-nets	http

Then, the routing table consequently contains the following default route:

Route #	Interface	Destination	Gateway	Metric	Monitoring
1	wan	0.0.0.0/0	195.66.77.34	10	Off

Now, you decide to add a secondary route over a DSL connection and enable route monitoring. Hence, the updated routing table will look like:

Route #	Interface	Destination	Gateway	Metric	Monitoring
1	wan	0.0.0.0/0	195.66.77.34	10	On
2	dsl	0.0.0.0/0	193.54.68.129	20	Off

As long as the preferred wan route is healthy, everything will work as expected. Route monitoring will also be working, so the secondary route will be enabled should the wan route fail.

There are some issues, however, with this setup: imagine that a route failover has occurred, and the default route is now using the **dsl** interface. Then a new HTTP connection is being established from the **intnet** network. A route lookup will be made, resulting in a destination interface of **dsl**. The IP rules will then be evaluated, but as our NAT rule assumed the destination interface to be **wan**, the new connection will be dropped by the ruleset.

In addition, any *existing* connections matching the NAT rule will also be dropped as a result of the change in destination interface. Clearly, this was not the intention.

To overcome this issue, you will need to group potential destination interface into an interface group, enable the *Security/Transport Equivalent* flag for the group, and finally use that group as destination interface in your policies. For more information on groups, see Section 3.3.7, "Interface Groups".

## 4.2.3. Proxy ARP

As explained previously in Section 3.4, “ARP”, the ARP protocol facilitates a mapping between an IP address and the MAC address of a node on an Ethernet network. However situations may exist where a network running Ethernet is separated into two parts with a routing device such as an installed Clavister Security Gateway, in between. In such a case, CorePlus itself can respond to ARP requests directed to the network on the other side of the Clavister Security Gateway using the feature known as Proxy ARP.

For example, host A on one subnet might send an ARP request to find out the MAC address of the IP address of host B on another separate network. The proxy ARP feature means that CorePlus responds to this ARP request instead of host B. The CorePlus sends it's own MAC address instead in reply, essentially pretending to be the target host. After receiving the reply, Host A then sends data directly to CorePlus which, acting as a proxy, forwards the data on to host B. In the process the device has the opportunity to examine and filter the data.

The splitting of an Ethernet network into two distinct parts is a common application of Clavister Security Gateway's Proxy ARP feature, where access between the parts needs to be controlled. In such a scenario CorePlus can monitor and regulate all traffic passing between the two parts.



### **Note**

*It is only possible to have Proxy ARP functioning for Ethernet and VLAN interfaces.*

## 4.3. Policy-based Routing

### 4.3.1. Overview

Policy-based Routing is an extension to the standard approach to routing described previously. It offers administrators significant flexibility in implementing routing decision policies.

Normal routing forwards packets according to destination IP address information derived from static routes or from a dynamic routing protocol. For example, using OSPF, the route chosen for packets will be the least-cost (shortest) path derived from an SPF calculation. Policy-based Routing means that the routes chosen for traffic can be based on various parameters, such as the source address or service type.

CorePlus implements routing by not only looking at packets one by one, but by implementing it on a state or connection basis so that routing policy provides control on both the forward and return routing directions.

Policy-based Routing can also be applied on an application basis by allowing:

<b>Source based routing</b>	When more than one ISP is used to provide Internet services, Policy-based Routing can route traffic originating from different sets of users through different routes. For example, traffic from one address range might be routed through one ISP, whilst traffic from another address range might be through a second ISP.
<b>Service-based routing</b>	Policy-based Routing can route a given protocols such as HTTP, through transparent proxies such as Web caches.
<b>Creating provider-independent metropolitan area networks</b>	All users share a common active backbone, but each can use different ISPs, subscribing to different providers.

Policy-based Routing implementation in CorePlus consists of two elements:

- One or more user-defined *alternate* routing tables in addition to the standard default **main** routing table.
- Policy-based routing rules or associations between routing tables and interfaces, which determines which routing table to use.

### 4.3.2. Policy-based Routing Tables

CorePlus, as standard, has one default routing table called **main**. In addition to the **main** table, it is possible to define one or more, additional *alternate* routing tables (this section will refer to Policy-based Routing Tables as *alternate* routing tables).

Alternate routing tables contain the same fields for describing routes as *main*, except that there is an extra parameter *ordering* defined for each of them. This parameter decides how route lookup is done with the alternate table in conjunction with the **main** table. It is described further below in Section 4.3.6, “The *Ordering* parameter”.

The total number of routing tables that can be created is limited only by the particular license that is used with a CorePlus installation. At minimum, there is a limit of 5 alternate tables in addition to the *main* table.

### 4.3.3. Policy-based Routing Rules

A rule in the Policy-based Routing rule-set can decide which routing table is selected. A Policy-based Routing rule can be triggered by the type of Service (eg. HTTP) in combination with the Source/Destination Interface and Source/Destination Network.

When looking up Policy-based Rules, it is the first matching rule found that is triggered.

### 4.3.4. Interface/Routing Table Association

If an alternate routing table is to be selected for a given interface regardless of the service then there is a way other than Policy-based Routing Rules for specifying that an alternate routing table is to be used. It is possible to associate a given source interface directly with a particular alternate routing table. This is done with the **Group** membership field of the interface.

The only difference in this method of association is that the administrator cannot specify for which service, such as HTTP, this will apply.

### 4.3.5. Policy-based Routing Table Selection

When a new connection is established the following is the way the decision is made as to which routing table is chosen:

1. Access Rules are checked first.
2. The source interface is looked up in the **main** routing table to find the packet's destination address.
3. A search is made for a Policy-based Routing rule that matches the source and destination interface. If a matching rule is found then this determines the routing table to use.
4. If no matching PBR rule is found then a check is made to see if the source interface has an alternate routing table associated with it. In other words, to see if the interface is a *member* of a routing table.
5. Once the correct routing table has been located, a final check is made to make sure that the source IP address *is* routed on the receiving interface. This is done by making a reverse lookup in the routing table using the source IP address. If this check fails then a **Default access rule** error message is generated.
6. The connection is then subject to the normal IP Rule-set. If a SAT rule is encountered, address translation will be performed. The decision of which routing table to use is made before carrying out address translation but the actual route lookup is performed on the altered address.
7. The packet is finally forwarded and the new connection opened by CorePlus.

### 4.3.6. The *Ordering* parameter

Once the routing table for a new connection is chosen and that table is an alternate routing table, the *Ordering* parameter associated with the table is used to decide how the Alternate Table is combined with the **main** table to lookup the appropriate route. The three options are:

1. **Default** - The default behaviour is to look up the connection's route in the **main** table and then the alternate table. If a matching route is found in the **alternate** then this has precedence over a matching route in *main*. In other words an **all-nets** route in the alternate would replace an **all-nets** route in main.
2. **First** - The behaviour here is to look up the connection's route in the alternate table and then the **main** table. If a matching route is not found in the alternate table then the matching route in

the **main** is used.

3. **Only** - This options ignores the existence of any other table except the alternate table. One application of this is to give the administrator a way to dedicate a single routing table to one set of interfaces.

The first two options can be regarded as combining the alternate table with the **main** table and assigning one route if there is a match in both tables.



### ***Important - Ensuring all-nets appears in the main table.***

*A common mistake with Policy-based routing is the absence of a route in the default main routing table with a destination interface of **all-nets**. The absence of an **all-nets** route will prevent both Policy-based Routing Rules from working as well as the alternative of having interfaces being associated with alternate routing tables.*

#### **Example 4.3. Creating a Policy-Based Routing table**

In this example we create a Policy-based Routing table named "TestPBRTTable".

##### ***Clavister FineTune***

1. Go to **Routing > Policy-Based Routing Tables**
2. Choose **New Policy-Based Routing Table** from the context menu, the **Policy-Based Routing Table Properties** dialog box will be displayed.
3. Now enter:
  - **Name:** TestPBRTTable
  - For **Ordering** select one of:
    - **First** - the named routing table is consulted first of all. If this lookup fails, the lookup will continue in the main routing table.
    - **Default** - the main routing table will be consulted first. If the only match is the default route (0.0.0.0/0), the named routing table will be consulted. If the lookup in the named routing table fails, the lookup as a whole is considered to have failed.
    - **Only** - the named routing table is the only one consulted. If this lookup fails, the lookup will not continue in the main routing table.
4. If **Remove Interface IP Routes** is enabled, the default interface routes are removed, i.e. routes to the *core* interface (which are routes to the Clavister Security Gateway itself).
5. Click **OK**.

#### **Example 4.4. Creating the Route**

After defining the routing table "TestPBRTTable", we add routes into the table.

##### ***Clavister FineTune***

1. Select the **Routing > Routing Tables > TestPBRTTable** section of the target system in the tree view of the Security Editor.
2. Choose **New Policy-Based Route...** from the context menu, the **Policy-Based Route Properties** dialog box will be displayed.
3. Now enter:
  - **Interface:** The interface to be routed
  - **Network:** The network to route



- **Gateway:** The gateway to send routed packets to
  - **Local IP Address:** The IP address specified here will be automatically published on the corresponding interface. This address will also be used as the sender address in ARP queries. If no address is specified, the gateway's interface IP address will be used.
  - **Metric:** Specifies the metric for this route. (Mostly used in route fail-over scenarios)
4. Click **OK**.

#### Example 4.5. Policy Based Routing Configuration

This example illustrates a multiple ISP scenario which is a common use of Policy-based Routing. The following is assumed:

- Each ISP will give you an IP network from its network range. We will assume a 2-ISP scenario, with the network 1.2.3.0/24 belonging to "ISP A" and "2.3.4.0/24" belonging to "ISP B". The ISP gateways are 1.2.3.1 and 2.3.4.1, respectively.
- All addresses in this scenario are public addresses for the sake of simplicity.
- This is a "drop-in" design, where there are no explicit routing subnets between the ISP gateways and the Clavister Security Gateway.

In a provider-independent metropolitan area network, clients will likely have a single IP address, belonging to one of the ISPs. In a single-organization scenario, publicly accessible servers will be configured with two separate IP addresses: one from each ISP. However, this difference does not matter for the policy routing setup itself.

Note that, for a single organization, Internet connectivity through multiple ISPs is normally best done with the BGP protocol, where you do not need to worry about different IP spans or policy routing. Unfortunately, this is not always possible, and this is where Policy Based Routing becomes a necessity.

We will set up the main routing table to use ISP A, and add a named routing table, "r2" that uses the default gateway of ISP B.

Interface	Network	Gateway	ProxyARP
lan1	1.2.3.0/24		wan1
lan1	2.3.4.0/24		wan1
wan1	1.2.3.1/32		wan1
wan2	2.3.4.1/32		lan1
wan1	0.0.0.0/0	1.2.3.1	

Contents of the named Policy-based Routing table r2:

Interface	Network	Gateway
wan2	0.0.0.0/0	2.3.4.1

The table r2 has its Ordering parameter set to Default, which means that it will only be consulted if the main routing table lookup matches the default route (0.0.0.0/0).

Contents of the Policy-based Routing Policy:

Source Interface	Source Range	Destination Interface	Destination Range	Service	Forward VR table	Return VR table
lan1	1.2.3.0/24	wan2	0.0.0.0/0	ALL	r2	r2
wan2	0.0.0.0/0	lan1	2.3.4.0/24	ALL	r2	r2

To configure this example scenario:

#### **Clavister FineTune**

1. Add the routes found in the list of routes in the main routing table, as shown earlier.
2. Create the Policy-based Routing table. Name the table "r2" and make sure the ordering is set to "Default".
3. Add the route found in the list of routes in the named policy routing table "r2", as shown earlier.
4. Add two VR policies according to the list of policies shown earlier.
  - Select the **Routing > Policy-Based Routing Rules** section of the target system in the tree view of the Security Editor.
  - Choose **New Policy-Based Routing Rule...** from the context menu, the **Policy-Based Routing Rule Properties** dialog box will be displayed.
  - Enter the information found in the list of policies displayed earlier.
  - Repeat the above to add the second rule.

**Note**

*Rules in the above example are added for both inbound and outbound connections.*

## 4.4. Virtual Routing

### 4.4.1. Overview

Virtual Routing is a logical construct that builds on old as well as new technology, enabling the creation of multiple, logically separated systems with their own routing tables and rulesets, communicating between them just as physically separate systems would.

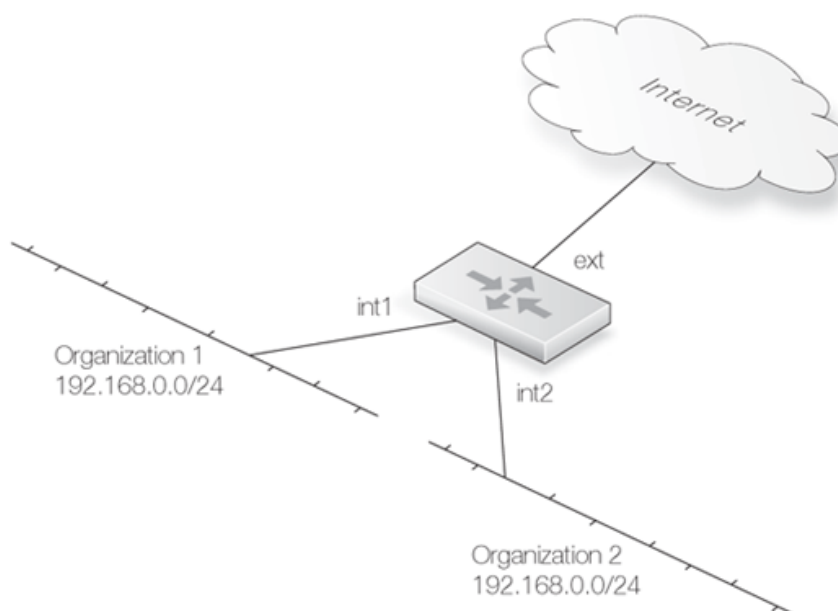
The basic building blocks are:

- Separate routing table for each virtual system
- Per-interface policy-based routing table membership - to make interface IP addresses reachable only in the desired routing table
- Pairs of loopback interfaces for inter-system communication

Nearly everything that can be done with physically separate units can also be done with virtual systems, including running dynamic routing processes such as OSPF on each system.

### 4.4.2. The Problem

Policy-based routing in and of its own can solve many of the problems that virtual systems can. However, very large configurations tend to become unwieldy as the number of all-to-all mappings grow. Consider for instance the problem of one physical unit firewalling two separate organizations, both using the same IP span.



This router would use two PBR routing tables, one for each organization:

**prbrtable1**

	Interface	Network	Gateway
1	ext	0.0.0.0/0	defaultgw
2	int1	192.168.0.0/24	

**prbrtable2**

	Interface	Network	Gateway
1	ext	0.0.0.0/0	defaultgw
2	int2	192.168.0.0/24	

Getting traffic from each network to and from the Internet is straightforward. Assuming only outbound traffic, it takes two PBR rules. Assuming that each organization has a public IP address which maps to servers on the respective networks, we handle outbound as well as inbound traffic with four rules:

	Name	Source if	Source Net	Dest Net	Fwd PBR	Ret PBR
1	org1-in	ext	0.0.0.0/0	pubip-org1	prbrtable1	prbrtable1
2	org1-out	int1	0.0.0.0/0	0.0.0./0	prbrtable1	prbrtable1
3	org2-in	ext	0.0.0.0/0	pubip-org2	prbrtable2	prbrtable2
4	org2-out	int2	0.0.0.0/0	0.0.0./0	prbrtable2	prbrtable2

This works as long as the organizations do not attempt to access each other's public resources. When that happens, we would need two more rules, before the other four.

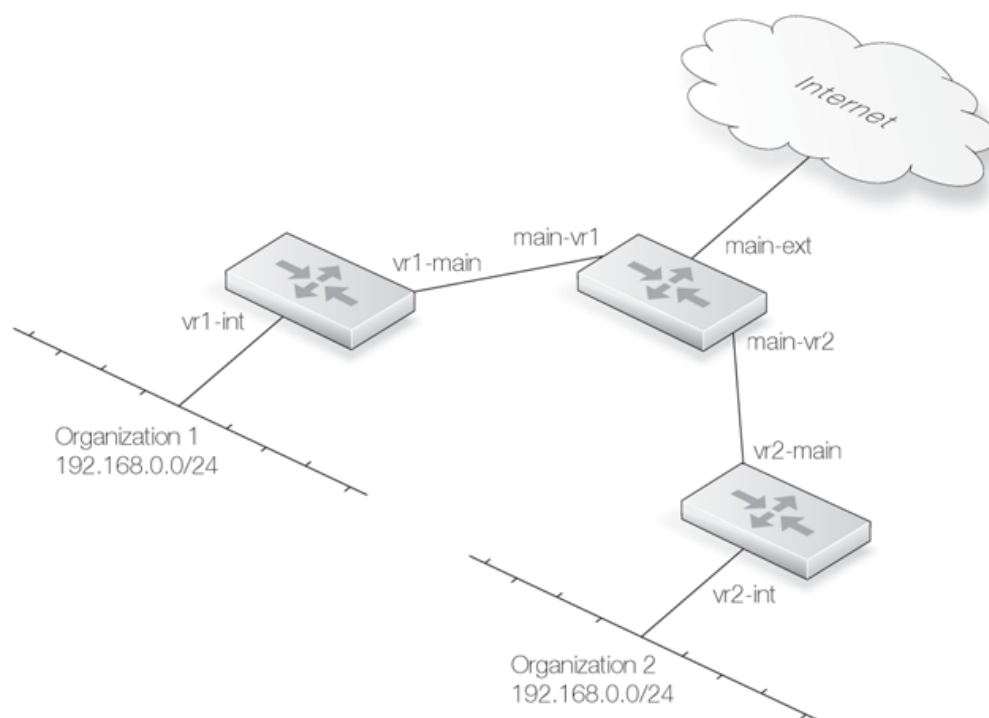
	Name	Source if	Source Net	Dest Net	Fwd PBR	Ret PBR
org1-org2	int1	0.0.0.0/0	pubip-org2	prbrtable2	prbrtable1	
org2-org1	int2	0.0.0.0/0	pubip-org1	prbrtable1	prbrtable2	

So, with two organizations, two rules are enough. However, with three organizations, six rules are needed. And with four organizations, twelve rules are needed. With five, twenty rules and the list continues: 30, 42, 56, 72, 90...

### 4.4.3. Benefits of Virtual Routing

One of the major benefits of Virtual Routing is eliminating the need for all-to-all mappings such as demonstrated above. Also, in using interface PBR table membership rather than PBR rules, the configuration complexity is reduced overall. There is also the additional benefit of making the gateway policy clearer by reducing the number of policy rules required the same way that the number of PBR rules is reduced.

Taking the previous network layout as an example, we would construct three virtual routers:



Each organization gets a virtual router of its own. Both connect to the "main" router, using pairs of loopback interfaces. First, we examine the routing tables:

#### main

	Interface	Network	Gateway
1	main-ext	0.0.0.0/0	defaultgw
2	main-vr1	pubip-vr1	
3	main-vr2	pubip-vr2	

#### vr1

	Interface	Network	Gateway
1	vr1-main	0.0.0.0/0	
2	vr1-int	192.168.0.0/24	

#### vr2

	Interface	Network	Gateway
1	vr2-main	0.0.0.0/0	
2	vr2-int	192.168.0.0/24	

Using per-interface PBR membership, we can leave the PBR ruleset empty. The contents of the Ethernet and Loopback interface sections follow:

#### Ethernet Interfaces

	Name	Driver	IP address	PBR table
1	main-ext	...	ip_main-ext	main
2	vr1-int	...	192.168.0.1	vr1
3	vr2-int	...	192.168.0.254	vr2

### Loopback Interfaces

	Interface	IP address	Loop to	PBR table
1	main-vr1	ip_main-ext	vr1-main	main
2	vr1-main	pubip-vr1	main-vr1	vr1
3	main-vr2	ip_main-ext	vr2-main	main
4	vr2-main	pubip-vr2	main-vr2	vr2

For each connection between a pair of virtual routers, two loopback interfaces are used; one in each virtual router. When a packet is sent through "main-vr1", it arrives as a newly received packet on "vr1-main". When a packet is sent through "vr1-main", it is received on "main-vr1". This is exactly the same as with two physically separate units: two interfaces, one in each, with a connection between them.

The PBR Table membership settings mean that if a connection arrives on an interface, it will be routed according to the PBR table that the interface is a member of.

Also make note of how the IP addresses of the internal interfaces of the virtual routers differ. If per-interface PBR table membership were not used, "core" routes for both IP addresses would be added in both routing tables, leading to 192.168.0.1 being unusable in vr2 even though it shouldn't be, and 192.168.0.254 being unusable in vr1. However, with per-interface PBR table membership, interface IP addresses belonging to one virtual router will not interfere with other virtual routers

The IP addresses of the main-vr1 and main-vr2 interfaces are the same as the IP address of the external interface. They could also have been set to something nonsensical, like 127.0.0.1. Regular routing would still have worked just fine since loopback interfaces are raw IP interfaces (the ARP protocol is not used over them). However, their IP addresses will be visible to users doing a traceroute from the inside, and also there is the issue of traffic originating from the gateway itself to the internal networks, such as pings or logging. Such traffic is most often routed according to the main routing table, and will be sourced from the IP address of the nearest interface in the main routing table

## 4.4.4. Policies in Virtual Systems

Policies in different virtual systems are not split up into different sections, simply because there is no need for it. Rather, policies for different virtual systems all reside in the regular ruleset. There are also benefits to this approach, for instance the possibility to define "shared" or "global" rules that span over several or *all* virtual systems. For instance, if an aggressive worm strikes, it may be desirable to drop all communication on ports known to be used by the worm until countermeasures can be put into place. One single drop rule placed at the top of the ruleset can take care of this for all virtual systems in one physical unit.

Using the previous example as a basis, we show how policies might look in virtual systems:

### Interface Groups

	Name	Interfaces
1	main-vrifs	main-vr1, main-vr2
2	main-ifs	main-ext, main-vrifs
3	vr1-ifs	vr1-main, vr1-int
4	vr2-ifs	vr2-main, vr2-int

## Rules

	Name	Action	Source if	Source Net	Dest if	Dest Net	Service
Rules for the "main" VR							
1	main-allwall	Allow	main-ifs	0.0.0.0/0	Any	0.0.0.0/0	All
Rules for the "vr1"							
2	vr1-http-in	SAT	vr1-ifs	0.0.0.0/0	pubip-vr1	0.0.0.0/0	http, SetDest 192.168.0.5
3	vr1-http-in	Allow	vr1-main	0.0.0.0/0	Any	pubip-vr1	http
4	vr1-out	NAT	vr1-int	0.0.0.0/0	Any	0.0.0.0/0	All
Rules for the "vr2"							
5	vr2-smtp-in	SAT	vr2-main	0.0.0.0/0	Any	pubip-vr2	All
6	vr2-smtp-in	Allow	vr2-main	0.0.0.0/0	Any	pubip-vr2	smtp
7	vr2-http-out	NAT	vr2-int	192.168.0.4	vr2-main	0.0.0.0/0	http

The security policies for the respective organizations are clearly very different, and this is all handled easily with a virtual system setup.

Note how the SAT rules do not need to take into account that there are more organizations connected to the same physical unit. There is no direct connection between them; everything arrives through the same interface, connected to the "main" VR. If this was done without virtual routing, the Allow rules would have to be preceded by NAT rules for traffic from other organizations. Care would also have to be taken that such rules were in accordance with the security policy of each organization. All such problems are eliminated with virtual systems.

The source interface filters are very specific. "Any" is not used as source interface anywhere, since such a rule would trigger regardless of the VR processing context. Consider for instance what would happen if the "vr1-http-in" rules were to use "Any" as source interface. They would trigger as soon as packets destined to pubip-vr1 were received on "main-ext". The destination address would be rewritten to 192.168.0.5, and passed on, using the main routing table. The main routing table would not know what to do with 192.168.0.5 and pass it back out to the default gateway outside the gateway.

If you use the same naming scheme as shown in this example, you can quickly make sure that the source interfaces are correct. All the rules concerning the main VR have source interfaces beginning with "main-". All the ones concerning vr1 have source interfaces beginning with "vr1-", and so on.

The destination interface filters, however, does not need to be as specific as the source interface filters. The possible destinations are limited by the routing tables used. If the "vr1" table only includes routes through "vr1-" interfaces, "Any" filters can only mean "through other interfaces in the same virtual router". It may however be a sound practice to write tighter destination interface filters in case an error has snuck into the configuration elsewhere. In this example, rule 1 might use "main-ifs", rule 4 might use "vr1-main". The SAT and corresponding Allow rules however are already fairly tight in that they only concern one single destination IP address.

## 4.4.5. Trouble Shooting

- Make sure that the source interface filters are correct

- Double check interface PBR table membership. For all types of interfaces and tunnels
- Use "ping -p <pbtable>" to source pings from different virtual routers
- Use "ping -r <rcvif> -s <srcip>" to test the ruleset, simulating that the ping was received on a given interface from a given IP address
- Use "arpsnoop -v <ifacenames>" to get verbose information about ARP resolution
- Use "route <pbtable> -all" to view all route entries in a given table, including "core" routes
- Use "route -lookup <ipaddr> <pbtable>" to make sure that a given IP address is routed the way you think in a given virtual router. (Hint: "-lookup" may be shortened to "-l".)
- Use "conn -v" to view verbose information about open connections. Both ends of a connection will be shown; before and after address translation. Also, the routing tables used in the forward and return direction will be shown.
- Enable logging and read the logs. In each virtual router, a separate rule decision is made, and a separate connection is established



## 4.5. Dynamic Routing

### 4.5.1. Dynamic Routing overview

Dynamic routing is different to static routing in that the Clavister Security Gateway will adapt to changes of network topology or traffic load automatically. CorePlus first learns of all the directly connected networks and gets further route information from other routers. Detected routes are sorted and the most suitable routes for destinations are added into the routing table and this information is distributed to other routers. Dynamic Routing responds to routing updates on the fly but has the disadvantage that it is more susceptible to certain problems such as routing loops. In the Internet, two types of dynamic routing algorithm are used: the Distance Vector(DV) algorithm and the Link State(LS) algorithm. How a router decides the optimal or "best" route and shares updated information with other routers depends on the type of algorithm used.

#### 4.5.1.1. Distance Vector algorithms

The Distance vector (DV) algorithm is a decentralized routing algorithm that computes the "best" path in a distributed way. Each router computes the costs of its own attached links, and shares the route information only with its neighbor routers. The router will gradually learn the least-cost path by iterative computation and information exchange with its neighbors.

The Routing Information Protocol (RIP) is a well-known DV algorithm and involves sending regular update messages and reflecting routing changes in the routing table. Path determination is based on the "length" of the path which is the number of intermediate routers {also known as "hops"}. After updating its own routing table, the router immediately begins transmitting its entire routing table to neighboring routers to inform them of changes.

#### 4.5.1.2. Link State algorithms

Different from the DV algorithms, Link State (LS) algorithms enable routers to keep routing tables that reflect the topology of the entire network. Each router broadcasts its attached links and link costs to all other routers in the network. When a router receives these broadcasts it runs the LS algorithm and calculates its own set of least-cost paths. Any change of the link state will be sent everywhere in the network, so that all routers keep the same routing table information.

Open Shortest Path First (OSPF) is a widely used LS algorithm. An OSPF enabled router first identifies the routers and subnets that are directly connected to it and then broadcasts the information to all the other routers. Each router uses the information it receives to build a table of what the whole network looks like. With a complete routing table, each router can identify the subnetworks and routers that lead to any destination. Routers using OSPF only broadcast updates that inform of changes and not the entire routing table.

OSPF depends on various metrics for path determination, including hops, bandwidth, load and delay. OSPF can provide a great deal of control over the routing process since its parameters can be finely tuned.

#### 4.5.1.3. Comparing dynamic routing algorithms

Due to the fact that the global link state information is maintained everywhere in a network, Link State algorithms have a high degree of configuration control and scalability. Changes result in broadcasts of just the updated information to other routers which results in faster convergence and less possibility of routing loops. OSPF can also operate within a hierarchy, whereas RIP has no knowledge of sub-network addressing. CorePlus uses OSPF as its dynamic routing algorithm because of the advantages it offers.

#### 4.5.1.4. Routing metrics

Routing metrics are the criteria a routing algorithm uses to compute the "best" route to a destination. A routing protocol relies on one or several metrics to evaluate links across a network and to determine the optimal path. The principal metrics used include:

<b>Path length</b>	The sum of the costs associated with each link. A commonly used value for this metric is called "hop count" which is the number of routing devices a packet must pass through when it travels from source to destination.
<b>Item Bandwidth</b>	The traffic capacity of a path, rated by "Mbps".
<b>Load</b>	The usage of a router. The usage can be evaluated by CPU utilization and throughput.
<b>Delay</b>	The time it takes to move a packet from the source to the destination. The time depends on various factors, including bandwidth, load, and the length of the path.

## 4.5.2. OSPF

### 4.5.2.1. OSPF Protocol Overview

Open Shortest Path First (OSPF) is a routing protocol developed for IP networks by the Internet Engineering Task Force (IETF). The CorePlus OSPF implementation is based upon RFC 2328, with compatibility to RFC 1583.

The way OSPF works is that it routes IP packets based only on the destination IP address found in the IP packet header. IP packets are routed "as is", that is they are not encapsulated in any further protocol headers as they transit the Autonomous System (AS). OSPF is a dynamic routing protocol, it quickly detects topological changes in the AS (such as router interface failures) and calculates new loop-free routes after a period of time.

OSPF is a link-state routing protocol that calls for the sending of link-state advertisements (LSAs) to all other routers within the same area. In a link-state routing protocol, each router maintains a database describing the Autonomous System's topology. This database is referred to as the link-state database. Each router in the same AS has an identical database. From the information in the link-state database, each router constructs a tree of shortest paths with itself as root. This shortest-path tree gives the route to each destination in the Autonomous System.

OSPF allows sets of networks to be grouped together, this is called an area. The topology of an area is hidden from the rest of the AS. This information hiding reduces the amount of routing traffic exchanged. Also, routing within the area is determined only by the area's own topology, lending the area protection from bad routing data. An area is a generalization of an IP subnetted network.

All OSPF protocol exchanges can be authenticated. This means that only routers with the correct authentication can join the Autonomous System. Different authentication schemes can be used, like none, passphrase or MD5 digest. It is possible to configure separate authentication methods for each Autonomous System.

### 4.5.2.2. OSPF Area Overview

The Autonomous System is divided into smaller parts called OSPF Areas. This chapter describes what an area is, and associated terms.

<b>Areas</b>	An area consists of networks and hosts within an AS that have been grouped together. Routers that are only within an area are called internal routers, all interfaces on internal routers are directly connected to networks within the area. The topology of an area is hidden from the rest of the AS.
<b>ABRs</b>	Routers that have interfaces in more than one area are called Area Border Routers (ABRs), these maintain a separate topological database for each area to which they have an interface.
<b>ASBRs</b>	Routers that exchange routing information with routers in other Autonomous Systems are called Autonomous System Boundary Router (ASBRs). They

advertise externally learned routes throughout the Autonomous System.

<b>Backbone Areas</b>	All OSPF networks need to have at least the backbone area, that is the area with ID 0. This is the area that all other areas should be connected to, and the backbone make sure to distribute routing information between the connected areas. When an area is not directly connected to the backbone it needs a virtual link to it.
<b>Stub Areas</b>	Stub areas are areas through which or into which AS external advertisements are not flooded. When an area is configured as a stub area, the router will automatically advertises a default route so that routers in the stub area can reach destinations outside the area.
<b>Transit Areas</b>	Transit areas are used to pass traffic from a area that is not directly connect to the backbone area.

### 4.5.2.3. Designated Router

Each OSPF broadcast network has a designated router and a backup designated router. The routers uses OSPF hello protocol to elect the DR and BDR for the network based on the priorities advertised by all the routers. If there already are a DR on the network, the router will accept that one, regardless of its own router priority.

### 4.5.2.4. Neighbors

Routers that are in the same area become neighbors in that area. Neighbors are elected via the Hello protocol. Hello packets are sent periodically out of each interface using IP multicast. Routers become neighbors as soon as they see themselves listed in the neighbor's Hello packet. This way, a two way communication is guaranteed.

The following *Neighbor States* are defined:

<b>Down</b>	This is the initial stat of the neighbor relationship.
<b>Init</b>	When a HELLO packet is received from a neighbor, but does NOT include the Router ID of the gateway in it, the neighbor will be placed in Init state. As soon as the neighbor in question receives a HELLO packet it will know the sending routers Router ID and will send a HELLO packet with that included. The state of the neighbors will change to <i>2-way</i> state.
<b>2-Way</b>	In this state the communication between the router and the neighbor is bi-directional. On Point-to-Point and Point-to-Multipoint interfaces, the state will be changed to <i>Full</i> . On Broadcast interfaces, only the DR/BDR will advance to <i>Full</i> state with their neighbors, all the remaining neighbors will remain in the <i>2-Way</i> state.
<b>ExStart</b>	Preparing to build adjacency.
<b>Exchange</b>	Routers are exchanging Data Descriptors.
<b>Loading</b>	Routers are exchanging LSAs.
<b>Full</b>	This is the normal state of an adjacency between a router and the DR/BDR.

### 4.5.2.5. Aggregates

OSPF Aggregation is used to combine groups of routes with common addresses into a single entry in the routing table. This is commonly used to minimize the routing table.

### 4.5.2.6. Virtual Links

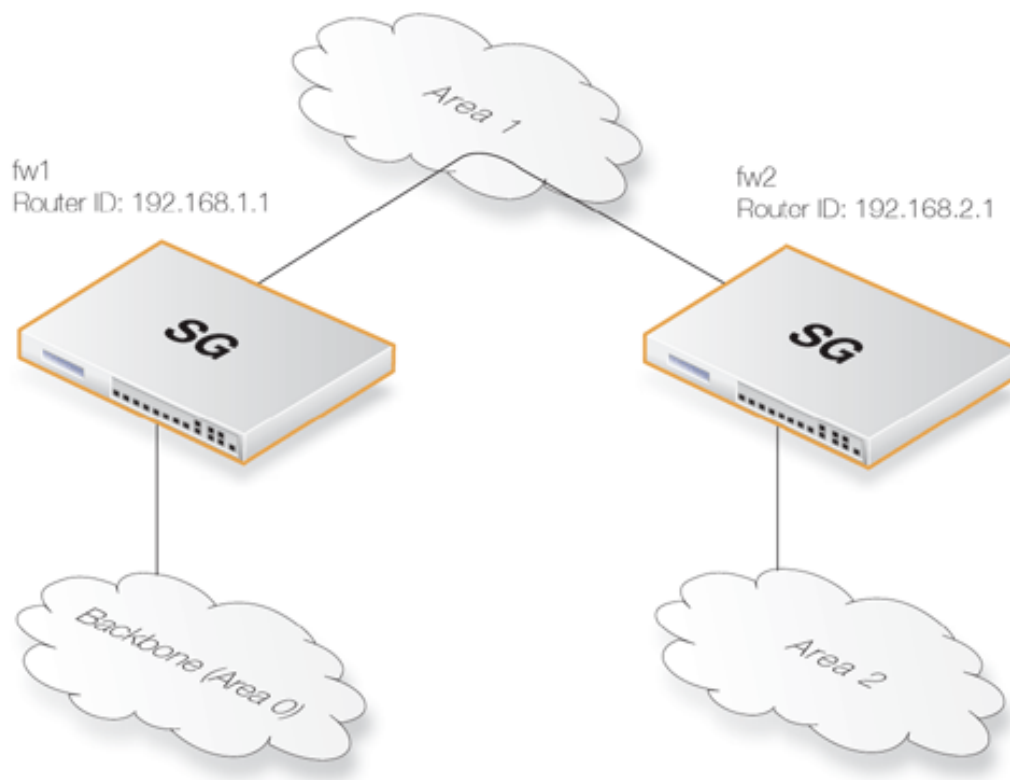
Virtual links are used for:

- Linking an area that does not have a direct connection to the backbone.
- Linking the backbone in case of a partitioned backbone.

### Area without direct connection to the backbone

The backbone always need to be the center of all other areas. In some rare case where it is impossible to have an area physically connected to the backbone, a virtual link is used. This virtual link will provide that area with a logical path to the backbone area. This virtual link is established between two ABRs that are on one common area, with one of the ABRs connected to the backbone area. In the example below two routers are connected to the same area (Area 1) but just one of them, fw1, is connected physically to the backbone area.

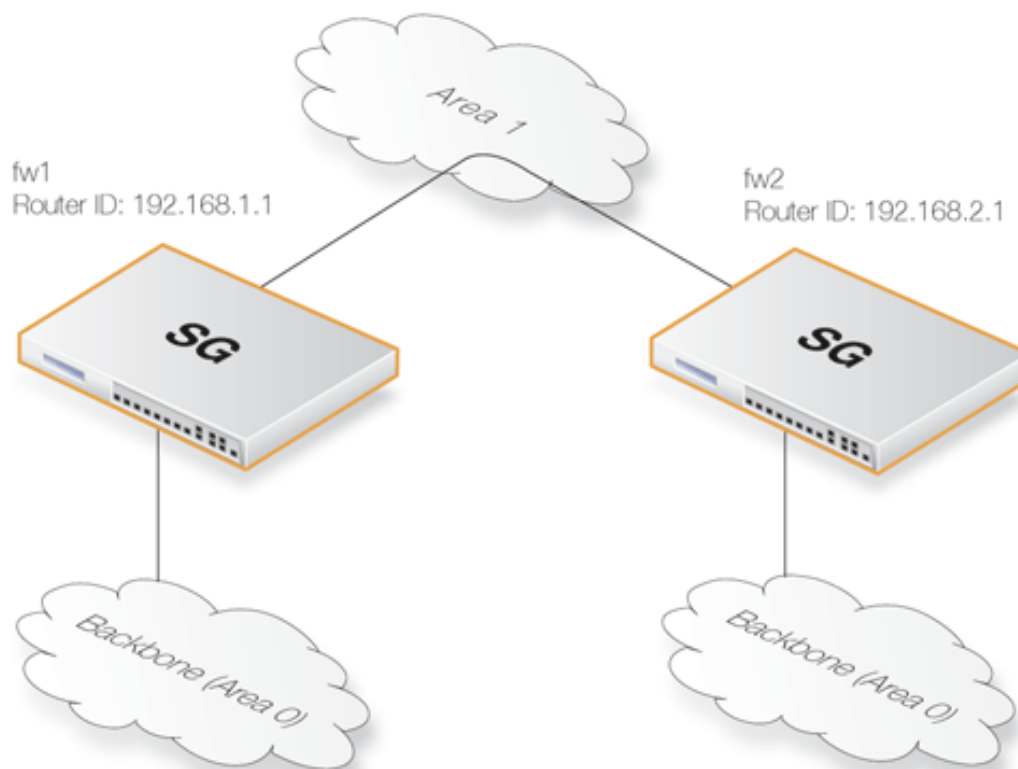
**Figure 4.2. Virtual Links Example 1**



In the above example, the Virtual Link is configured between fw1 and fw2 on Area 1, as it is used as the transit area. In this configuration only the Router ID has to be configured. The diagram shows that fw2 needs to have a Virtual Link to fw1 with Router ID 192.168.1.1 and vice versa. These Virtual Links need to be configured in Area 1.

### Partitioned Backbone

OSPF allows for linking a partitioned backbone using a virtual link. The virtual link should be configured between two separate ABRs that touch the backbone are from each side and having a common area in between.

**Figure 4.3. Virtual Links Example 2**

The Virtual Link is configured between fw1 and fw2 on Area 1, as it is used as the transit area. In the configuration only the Router ID have to be configured, as in the example above show fw2 need to have a Virtual Link to fw1 with the Router ID 192.168.1.1 and vice versa. These VLinks need to be configured in Area 1.

### 4.5.2.7. OSPF High Availability Support

There are some limitations in HA support for OSPF that should be noted:

Both the active and the inactive part of an HA cluster will run separate OSPF processes, although the inactive part will make sure that it is not the preferred choice for routing. The HA master and slave will not form adjacency with each other and are not allowed to become DR/BDR on broadcast networks. This is done by forcing the router priority to 0.

For OSPF HA support to work correctly, the gateway needs to have a broadcast interface with at least ONE neighbor for ALL areas that the gateway is attached to. In essence, the inactive part of the cluster needs a neighbor to get the link state database from.

It should also be noted that is not possible to put two HA gateways on the same broadcast network without any other neighbors (they won't form adjacency with each other because of the router priority 0). However it, based on scenario, may be possible to setup a point to point link between them instead. Special care must also be taken when setting up a virtual link to an HA gateway. The end-point setting up a link to the HA gateway must setup 3 separate links: one to the shared, one the master and one to the slave router id of the gateway.

## 4.5.3. Dynamic Routing Policy

### 4.5.3.1. Overview

In a dynamic routing environment, it is important for routers to be able to regulate to what extent they will participate in the routing exchange. It is not feasible to accept or trust all received routing information, and it might be crucial to avoid that parts of the routing database gets published to other routers.

For this reason, CorePlus provides a *Dynamic Routing Policy*, which is used to regulate the flow of dynamic routing information.

A Dynamic Routing Policy rule filters either statically configured or OSPF learned routes according to parameters like the origin of the routes, destination, metric and so forth. The matched routes can be controlled by actions to be either exported to OSPF processes or to be added to one or more routing tables.

The most common usages of Dynamic Routing Policy are:

- Importing OSPF routes from an OSPF process into a routing table.
- Exporting routes from a routing table to an OSPF process.
- Exporting routes from one OSPF process to another.



### Note

By default, CorePlus will not import or export any routes. In other words, for dynamic routing to be meaningful, it is mandatory to define at least one Dynamic Routing Policy rule.

## 4.5.3.2. Importing Routes from an OSPF AS into the Main Routing Table

In this example, the routes received using OSPF will be added into the main routing table.

### Example 4.6. Importing Routes from an OSPF AS into the Main Routing Table

First of all a Dynamic Routing Policy filter needs to be created. The filter needs to have a name, in this example *ImportOSPFRoutes* is used, as it explains what the filter does.

The filter must also specify from what OSPF AS the routes should be imported. In this example, a pre-configured OSPF AS named *as0* is used.

Depending on how your routing topology looks like you might want to just import certain routes using the *Destination Interface/Destination Network* filters, but in this scenario all routes that are within the *all-nets* network (i.e. 0.0.0.0/0) are allowed.

#### Clavister FineTune

1. Select the **Routing > Dynamic Routing Policy** section of the target system in the tree view of the Security Editor.
2. Choose **New Policy Filter...** from the context menu, the **Dynamic Routing Policy Filter Properties** dialog box will be displayed.
3. Specify a suitable name for the filter, for instance *ImportOSPFRoutes*.
4. Choose *as0* in the **From OSPF AS** control.
5. Choose *all-nets* in the **...Exactly Matches** control.
6. Click **OK**.

The next step is to create a Dynamic Routing Action that will do the actual importing of the routes into a routing table. Specify the destination routing table that the routes should be added to, in this case *main*.

**Clavister FineTune**

1. Select the **Routing > Dynamic Routing Policy > ImportOSPFRoutes** section of the target system in the tree view of the Security Editor.
2. Choose **New Routing Action...** from the context menu, the **Routing Action Properties** dialog box will be displayed.
3. In the **Destination** control, select **main**.
4. Click **OK**.

### 4.5.3.3. Exporting the Default Route into an OSPF AS

In this example, the default route from the main routing table will be exported into an OSPF AS named as0.

**Example 4.7. Exporting the Default Route into an OSPF AS**

Add a dynamic routing policy filter that matches the main routing table and the default route:

**Clavister FineTune**

1. Select the **Routing > Dynamic Routing Policy** section of the target system in the tree view of the Security Editor.
2. Choose **New Policy Filter...** from the context menu, the **Dynamic Routing Policy Filter Properties** dialog box will be displayed.
3. Specify a suitable name for the filter, for instance **ExportDefRoute**.
4. Choose **main** in the **From Routing Table** control.
5. Choose **wan** in the **Destination Interface** control.
6. Choose **all-nets** in the **...Exactly Matches** control.
7. Click **OK**.

Then, create an OSPF Action that will export the filtered route to the specified OSPF AS:

**Clavister FineTune**

1. Select the **Routing > Dynamic Routing Policy > ExportDefRoute** section of the target system in the tree view of the Security Editor.
2. Choose **New OSPF Action...** from the context menu, the **OSPF Action Properties** dialog box will be displayed.
3. In the **Destination** control, select **as0**.
4. Click **OK**.

## 4.6. Multicast Routing

### 4.6.1. Overview

Certain types of internet interactions, such as conferencing and video broadcasts, require a single client or host to send the same packet to multiple receivers. This could be achieved through the sender duplicating the packet with different receiving IP addresses or by a broadcast of the packet across the internet. These solutions waste large amounts of sender resources or network bandwidth and are therefore not satisfactory. An appropriate solution should also be able to scale to large numbers of receivers.

Multicast Routing solves the problem by the network routers themselves, replicating and forwarding packets via the optimum route to all members of a group. The IETF standards that enable Multicast Routing are:

1. Class D of the IP address space which is reserved for multicast traffic. Each multicast IP address represent an arbitrary group of recipients.
2. The Internet Group Membership Protocol (IGMP) allows a receiver to tell the network that it is a member of a particular multicast group.
3. Protocol Independent Multicast (PIM) is a group of routing protocols for deciding the optimal path for multicast packets.

Multicast routing operates on the principle that an interested receiver joins a group for a multicast by using the IGMP protocol. PIM routers can then duplicate and forward packets to all members of such a multicast group, thus creating a *distribution tree* for packet flow. Rather than acquiring new network information, PIM uses the routing information from existing protocols, such as OSPF, to decide the optimal path.

A key mechanism in the Multicast Routing process is that of *Reverse Path Forwarding*. For unicast traffic a router is concerned only with a packet's destination. With multicast, the router is also concerned with a packets source since it forwards the packet on paths which are known to be downstream, away from the packet's source. This approach is adopted to avoid loops in the distribution tree.

By default multicast packets are routed by CorePlus to the **core** interface. SAT Multiplex rules are set up in the IP rule-set in order to perform forwarding to the correct interfaces. This is demonstrated in the examples which follow.



#### **Note**

*For multicast to function with an Ethernet interface on any Clavister Security Gateway, that interface must have multicast handling set to **On** or **Auto**. For further details on this see Section 3.3.2.1, "Ethernet Interface Basics".*

### 4.6.2. Multicast Forwarding using the SAT Multiplex Rule

The SAT Multiplex rule is used to achieve duplication and forwarding of packets through more than one interface. This feature implements multicast forwarding in CorePlus, where a multicast packet is sent through several interfaces. Note that, since this rule overrides the normal routing tables, packets that should be duplicated by the multiplex rule needs to be routed to the **core** interface.

By default, the multicast IP range **224.0.0.0/4** is always routed to **core** and does not have to be manually added to the routing tables. Each specified output interface can individually be configured with static address translation of the destination address. The **Interface** field in the **Interface/Net Tuple** dialog might be left empty if the **IPAddress** field is set. In this case, the output interface will



be determined by a route lookup on the specified IP address.

The multiplex rule can operate in one of two modes:

**Use IGMP** The traffic flow specified by the multiplex rule must have been requested by hosts using IGMP before any multicast packets are forwarded through the specified interfaces. This is the default behaviour of CorePlus.

**Not using IGMP** The traffic flow will be forwarded according to the specified interfaces directly without any inference from IGMP.



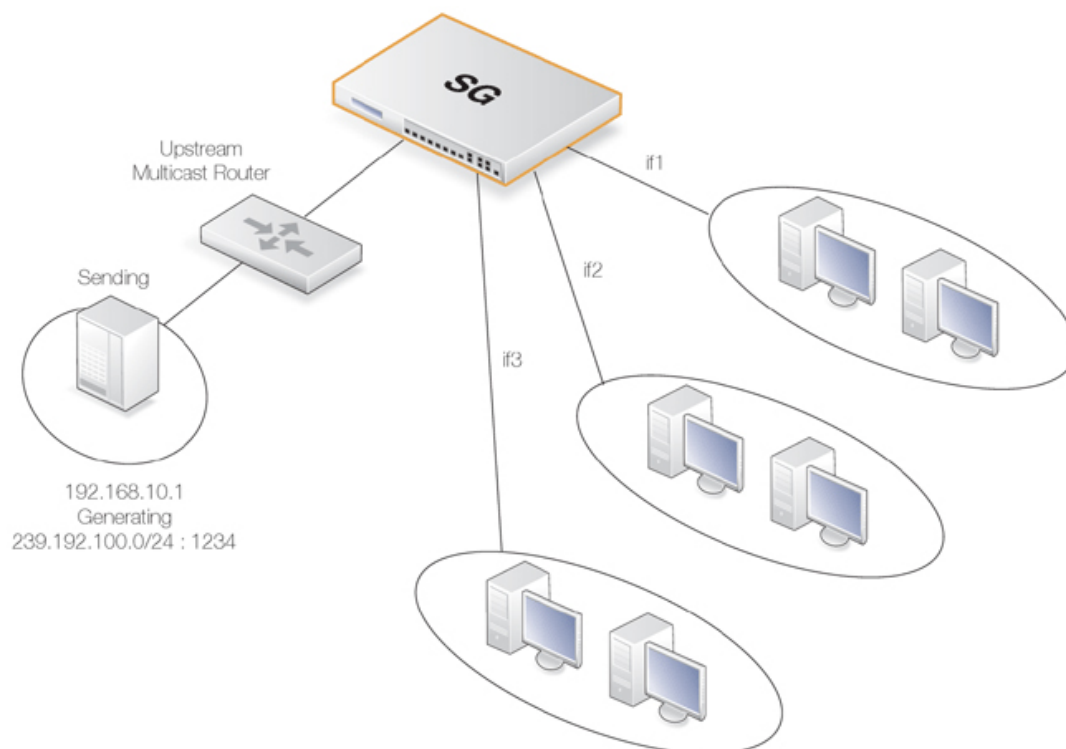
**Note**

Since the Multiplex rule is a SAT rule, an Allow or NAT rule has to be specified together with the Multiplex rule.

### 4.6.2.1. Multicast Forwarding - No Address Translation

This scenario describes how to configure multicast forwarding together with IGMP. The multicast sender is **192.168.10.1** and generates the multicast streams **239.192.10.0/24:1234**. These multicast streams should be forwarded from interface wan through the interfaces if1, if2 and if3. The streams should only be forwarded if some host has requested the streams using the IGMP protocol. The example below only covers the multicast forwarding part of the configuration. The IGMP configuration can be found below in Section 4.6.3.1, “IGMP Rules Configuration - No Address Translation”.

**Figure 4.4. Multicast Forwarding - No Address Translation**



**Note**

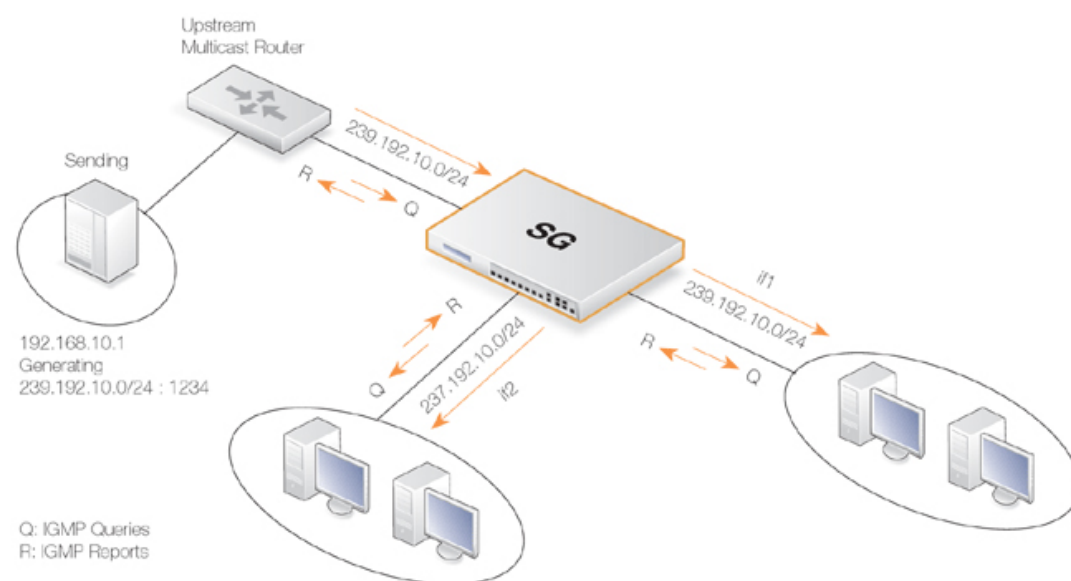
Remember to add an Allow rule matching the SAT Multiplex rule.

**Example 4.8. Forwarding of Multicast Traffic using the SAT Multiplex Rule**

In this example, we will create a multiplex rule in order to forward the multicast groups **239.192.10.0/24:1234** to the interfaces if1, if2 and if3. All groups have the same sender **192.168.10.1** which is located somewhere behind the wan interface. The multicast groups should only be forwarded to the out interfaces if clients behind those interfaces have requested the groups using IGMP. The following steps need to be executed to configure the actual forwarding of the multicast traffic. IGMP has to be configured separately.

**Clavister FineTune**

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu. The **Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance **Multicast\_Multiplex**.
4. Select **Multiplex\_SAT** in the **Action** dropdown list.
5. Select the following from the **Address Filter** dropdown lists:
  - **Source Interface:** wan
  - **Source Network:** 192.168.10.1
  - **Destination Interface:** core
  - **Destination Network:** 239.192.10.0/24
6. Under the **Service** tab, select **Custom** and **UDP** in the **Type** dropdown list. Specify port 1234 in the **Destination Port** field.
7. Under the **Address Translation** tab, click on the plus sign and add the output interfaces if1, if2 and if3 one at a time. For each interface, leave the **IPAddress** field blank since no destination address translation is wanted.
8. Make sure the **USE IGMP** checkbox is set.
9. Click **OK**.

**4.6.2.2. Multicast Forwarding - Address Translation Scenario****Figure 4.5. Multicast Forwarding - Address Translation**

This scenario is based on the previous scenario but now we are going to translate the multicast group. When the multicast streams **239.192.10.0/24** are forwarded through the if2 interface, the multicast groups should be translated into **237.192.10.0/24**. No address translation should be made when forwarding through interface if1. The configuration of the corresponding IGMP rules can be found below in Section 4.6.3.2, “IGMP Rules Configuration - Address Translation”.



### Caution

*As previously noted, remember to add an Allow rule matching the SAT Multiplex rule.*

#### Example 4.9. Multicast Forwarding - Address Translation

The following SAT Multiplex rule needs to be configured to match the scenario described above:

##### Clavister FineTune

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu. The **Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance **Multicast\_Multiplex**.
4. Select **Multiplex\_SAT** in the **Action** dropdown list.
5. Select the following from the **Address Filter** dropdown lists:
  - **Source Interface:** wan
  - **Source Network:** 192.168.10.1
  - **Destination Interface:** core
  - **Destination Network:** 239.192.10.0/24
6. Under the **Service** tab, select **Custom** and **UDP** in the **Type** dropdown list. Specify port 1234 in the **Destination Port** field.
7. Under the **Address Translation** tab, click on the plus sign. The **Interface/Net Tuple** dialog box will be displayed.
8. Add interface if1. Leave the **IPAddress** field empty and click OK.
9. Click on the plus sign again and add interface if2. This time, enter 237.192.10.0 in the **IPAddress** field and click OK.
10. Make sure the **USE IGMP** checkbox is set.
11. Click **OK**.



### Note

*If address translation of the source address is required, the **Allow** rule following the **SAT Multiplex** rule should be replaced with a **NAT** rule.*

## 4.6.3. IGMP Configuration

IGMP signalling between hosts and routers can be divided into two categories:

**IGMP Reports** Reports are sent from hosts towards the router when a host wants to subscribe to new multicast groups or change current multicast subscriptions.

**IGMP Queries** Queries are IGMP messages sent from the router towards the hosts in order to make sure that it will not close any stream that some host still wants to receive.

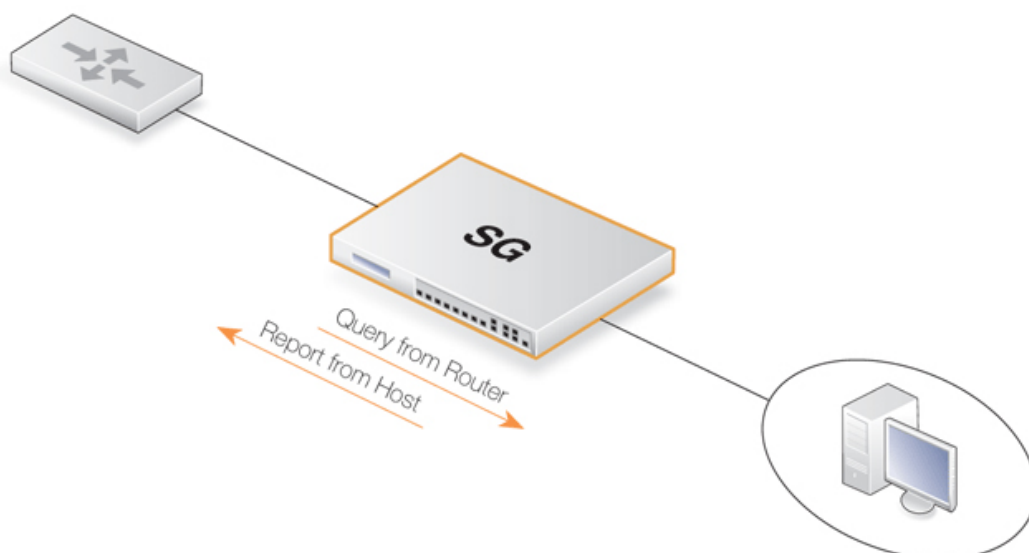
Normally, both these types of rules has to be specified for IGMP to work. One exception to this is if

the multicast source is located on a network directly connected to the router. In this case, no query rule is needed.

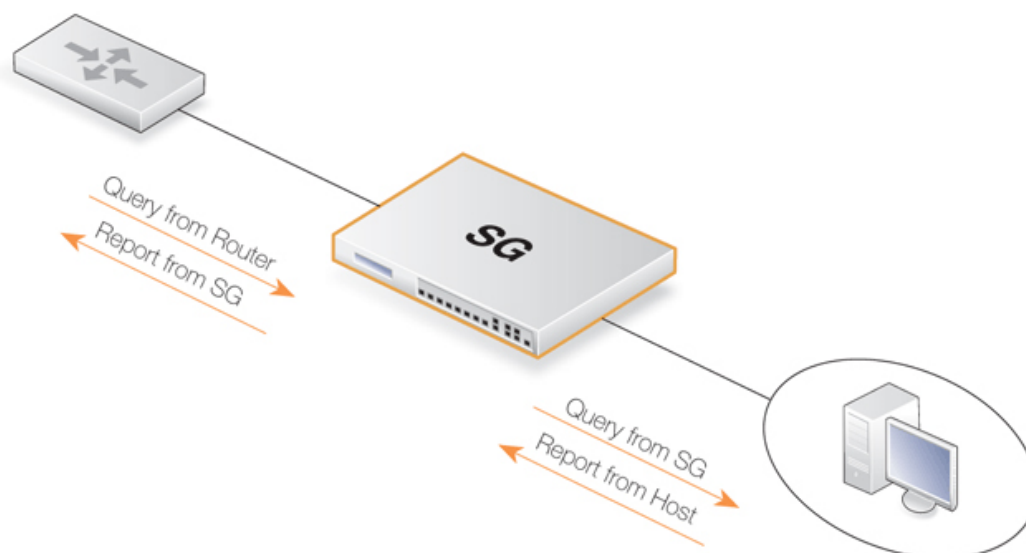
A second exception is if a neighbouring routing is statically configured to deliver a multicast stream to the Clavister Security Gateway. In this case also, an IGMP query would not have to be specified.

CorePlus supports two IGMP modes of operation - Snoop and Proxy.

**Figure 4.6. Multicast Snoop**



**Figure 4.7. Multicast Proxy**



In Snoop mode, the router will act transparently between the hosts and another IGMP router. It will not send any IGMP Queries. It will only forward queries and reports between the other router and the hosts. In Proxy mode, the router will act as an IGMP router towards the clients and actively send queries. Towards the upstream router, it will be acting as a normal host, subscribing to multicast groups on behalf of its clients.

### 4.6.3.1. IGMP Rules Configuration - No Address Translation

This example describes the IGMP rules needed for configuring IGMP according to the No Address Translation scenario described above. We want our router to act as a host towards the upstream router and therefore we configure IGMP to run in proxy mode.

#### Example 4.10. IGMP - No Address Translation

The following example requires a configured interface group IfGrpClients including interfaces if1, if2 and if3. The ip address of the upstream IGMP router is known as UpstreamRouterIP. Two rules are needed. The first one is a report rule that allows the clients behind interfaces if1, if2 and if3 to subscribe for the multicast groups **239.192.10.0/24**. The second rule, is a query rule that allows the upstream router to query us for the multicast groups that the clients have requested. The following steps need to be executed to create the two rules.

##### *Clavister FineTune*

1. Select the **Routing** section of the target system in the tree view of the Security Editor and select the **Multicast** and the **IGMP Rules** section.
2. Choose **New IGMP Rule...** from the context menu. The **IGMP Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance Reports.
4. Enter following:
  - **Type:** Report
  - **Action:** Proxy
  - **Relay Interface:** wan
  - **Source Interface:** IfGrpClients
  - **Source Network:** if1net, if2net, if3net
  - **Destination Interface:** core
  - **Destination Network:** auto
  - **Multicast Source:** 192.168.10.1
  - **Multicast Group:** 239.192.10.0/24
5. Click **OK**.
6. Choose **New IGMP Rule...** from the context menu. The **IGMP Rule Properties** dialog box will be displayed.
7. Specify a suitable name for the rule, for instance Queries.
8. Enter following:
  - **Type:** Query
  - **Action:** Proxy
  - **Relay Interface:** IfGrpClients
  - **Source Interface:** wan
  - **Source Network:** UpstreamRouterIp
  - **Destination Interface:** core
  - **Destination Network:** auto
  - **Multicast Source:** 192.168.10.1
  - **Multicast Group:** 239.192.10.0/24
9. Click **OK**.

### 4.6.3.2. IGMP Rules Configuration - Address Translation

The following examples illustrates the IGMP rules needed to configure IGMP according to the Address Translation scenario described above in Section 4.6.2.2, “Multicast Forwarding - Address Translation Scenario”. We need two IGMP report rules, one for each client interface. If1 uses no address translation and if2 translates the multicast group to **237.192.10.0/24**. We also need two query rules, one for the translated address and interface, and one for the original address towards if1. Two examples are provided, one for each pair of report and query rule. The upstream multicast router uses IP UpstreamRouterIP.

#### Example 4.11. Configuration if1

The following steps needs to be executed to create the report and query rule pair for if1 which uses no address translation.

##### *Clavister FineTune*

1. Select the **Routing** section of the target system in the tree view of the Security Editor and select the **Multicast** and the **IGMP Rules** section.
2. Choose **New IGMP Rule...** from the context menu. The **IGMP Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance Reports\_if1.
4. Enter following:
  - **Type:** Report
  - **Action:** Proxy
  - **Relay Interface:** wan
  - **Source Interface:** if1
  - **Source Network:** if1net
  - **Destination Interface:** core
  - **Destination Network:** auto
  - **Multicast Source:** 192.168.10.1
  - **Multicast Group:** 239.192.10.0/24
5. Click **OK**.
6. Choose **New IGMP Rule...** from the context menu. The **IGMP Rule Properties** dialog box will be displayed.
7. Specify a suitable name for the rule, for instance Queries\_if1.
8. Enter following:
  - **Type:** Query
  - **Action:** Proxy
  - **Relay Interface:** if1
  - **Source Interface:** wan
  - **Source Network:** UpstreamRouterIp
  - **Destination Interface:** core
  - **Destination Network:** auto
  - **Multicast Source:** 192.168.10.1
  - **Multicast Group:** 239.192.10.0/24
9. Click **OK**.

**Example 4.12. Configuration if2 - Group Translation**

The following steps need to be executed to create the report and query rule pair for if2 which translates the multicast group. Note that the group translated therefore the IGMP reports include the translated IP addresses and the queries will contain the original IP addresses.

**Clavister FineTune**

1. Choose **New IGMP Rule...** from the context menu. The **IGMP Rule Properties** dialog box will be displayed.
2. Specify a suitable name for the rule, for instance Reports\_if2.
3. Enter following:
  - **Type:** Report
  - **Action:** Proxy
  - **Relay Interface:** wan
  - **Source Interface:** if2
  - **Source Network:** if2net
  - **Destination Interface:** core
  - **Destination Network:** auto
  - **Multicast Source:** 192.168.10.1
  - **Multicast Group:** 237.192.100.0/24 (note: this is the translated group IP)
4. Under the **SAT** tab, enable **Translate the Multicast Group** and enter 239.192.10.0 in the **New IP Address** field.
5. Click **OK**.
6. Choose **New IGMP Rule...** from the context menu. The **IGMP Rule Properties** dialog box will be displayed.
7. Specify a suitable name for the rule, for instance Queries\_if2.
8. Enter following:
  - **Type:** Query
  - **Action:** Proxy
  - **Relay Interface:** if2
  - **Source Interface:** wan
  - **Source Network:** UpStreamRouterIP
  - **Destination Interface:** core
  - **Destination Network:** auto
  - **Multicast Source:** 192.168.10.1
  - **Multicast Group:** 239.192.10.0/24
9. Under the **SAT** tab, enable **Translate the Multicast Group** and enter 237.192.10.0 in the **New IP Address** field.
10. Click **OK**.

**Advanced IGMP Settings**

*There are a number of advanced settings which are global and apply to all interfaces which do not have IGMP settings explicitly specified for them. These global settings can be found in the advanced settings section of Clavister FineTune. Individual interface IGMP settings are found in the IGMP section of Clavister FineTune.*

## 4.7. Transparent Mode

### 4.7.1. Overview of Transparent Mode

Deploying Clavister Security Gateways operating in Transparent Mode into an existing network topology can significantly strengthen security. It's simple to do and doesn't require reconfiguration of existing nodes. Once deployed, CorePlus can then allow or deny access to different types of services (eg. HTTP) and in specified directions. As long as users of the network are accessing permitted services through the Clavister Security Gateway they are not aware of its presence. Transparent Mode is enabled by specifying a **Switch Route** instead of a standard **Route**.

A typical example of Transparent Mode's ability to improve security is in a corporate environment where there might be a need to protect different departments from one another. The finance department might require access to only a restricted set of services (eg. HTTP) on the sales department's servers whilst the sales department might require access to a similarly restricted set of applications on the finance department's network. By deploying a single Clavister Security Gateway between the two department's networks, transparent but controlled access can be achieved using the Transparent Mode feature.

Another example might be an organisation allowing traffic between the external internet and a range of public IP address' on an internal network. Transparent mode can control what kind of service is permitted to these IP addresses and in what direction. For instance the only services permitted in such a situation may be HTTP access out to the internet.

### 4.7.2. Comparison with Routing mode

The Clavister Security Gateway can operate in two modes: Routing Mode or Transparent Mode. In Routing Mode, the Clavister Security Gateway performs all the functions of a Layer 3 router; if the gateway is placed into a network for the first time, or if network topology changes, the routing configuration must therefore be thoroughly checked to ensure that the routing table is consistent with the new layout. Reconfiguration of IP settings may be required for pre-existing routers and protected servers. This mode works well when complete control over routing is desired.

In Transparent Mode, where **Switch Route** is used instead of **Route**, the gateway acts in a way that has similarities to a switch; it screens IP packets and forwards them transparently to the correct interface without modifying any of the source or destination information on the IP or Ethernet levels. Two benefits of Transparent Mode are:

- When a client moves from one interface to another without changing IP address, it can still obtain the same services as before (eg. HTTP, FTP) without routing reconfiguration.
- The same network address range can exist on several interfaces.



#### **Note**

*Clavister Security Gateways need not operate exclusively in Transparent Mode but can combine Transparent Mode with Routing Mode to operate in a hybrid mode. That is to say, the gateway can have both **Switch Routes** as well as standard routes defined. It is also possible to create a hybrid case by applying address translation on otherwise transparent traffic.*

### 4.7.3. Transparent Mode implementation

In transparent mode, CorePlus allows ARP transactions to pass through the Clavister Security Gateway, and determines from this ARP traffic the relationship between IP addresses, physical addresses and interfaces. CorePlus remembers this address information in order to relay IP packets to the correct receiver. During the ARP transactions, neither of the endpoints will be aware of the gateway's presence.



When beginning communication, a host will locate the target host's physical address by broadcasting an ARP request. This request is intercepted by CorePlus and it sets up an internal ARP Transaction State entry and broadcasts the ARP request to all the other switch-route interfaces except the interface the ARP request was received on. If CorePlus receives an ARP reply from the destination within a configurable timeout period, it will relay the reply back to the sender of the request, using the information previously stored in the ARP Transaction State entry.

During the ARP transaction, CorePlus learns the source address information for both ends from the request and reply. CorePlus maintains two tables to store this information: the Content Addressable Memory (CAM) and Layer 3 Cache. The CAM table tracks the MAC addresses available on a given interface and the Layer 3 cache maps an IP address to MAC address and interface. As the Layer 3 Cache is only used for IP traffic, Layer 3 Cache entries are stored as single host entries in the routing table.

For each IP packet that passes through the gateway, a route lookup for the destination is done. If the route of the packet matches a **Switch Route** or a Layer 3 Cache entry in the routing table, CorePlus knows that it should handle this packet in a transparent manner. If a destination interface and MAC address is available in the route, CorePlus has the necessary information to forward the packet to the destination. If the route was a **Switch Route**, no specific information about the destination is available and the gateway will have to discover where the destination is located in the network. Discovery is done by CorePlus sending out ARP as well as ICMP (ping) requests, acting as the initiating sender of the original IP packet for the destination on the interfaces specified in the **Switch Route**. If an ARP reply is received, CorePlus will update the CAM table and Layer 3 Cache and forward the packet to the destination.

If the CAM table or the Layer 3 Cache is full, the tables are partially flushed automatically. Using the discovery mechanism of sending ARP and ICMP requests, CorePlus will rediscover destinations that may have been flushed.

## 4.7.4. Enabling Transparent Mode

Two steps are normally required to have CorePlus operate in Transparent Mode:

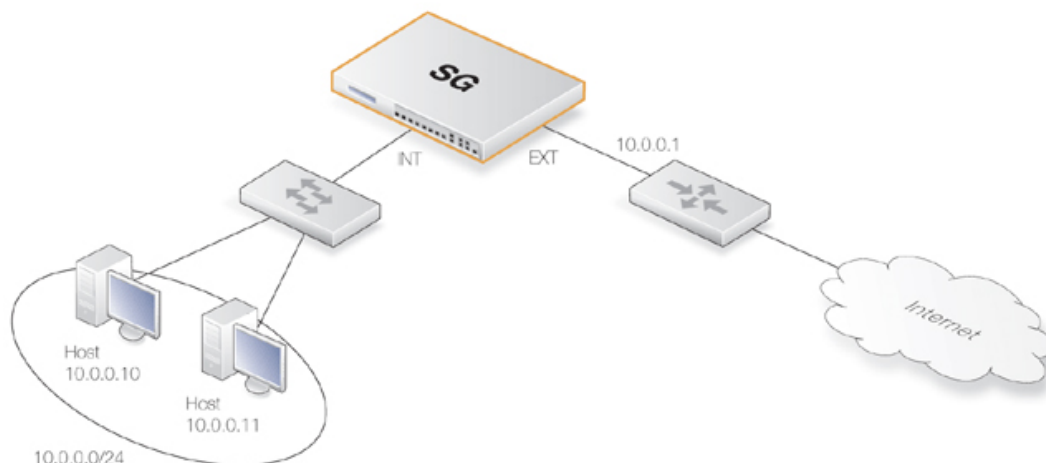
1. If desired, create a group of the interfaces that are to be transparent. Interfaces in a group can be marked as **Security transport equivalent** if hosts are to move freely between them.
2. Create **Switch Routes** and if applicable use the interface group created earlier. For the **Network** parameter, specify the range of IP addresses that will be transparent between the interfaces. *When the entire gateway is working in Transparent Mode this is range is normally 0.0.0.0/0.*

## 4.7.5. Transparent Mode example scenarios

### Scenario 1

The gateway in Transparent Mode is placed between an Internet access router and the internal network. The router is used to share the Internet connection with a single public IP address. The internal NAT:ed network behind the gateway is in the 10.0.0.0/24 address space. Clients on the internal network are allowed to access the Internet via the HTTP protocol.

**Figure 4.8. Transparent mode scenario 1**



### Example 4.13. Setting up Transparent Mode - Scenario 1

#### *Clavister FineTune*

Configure the interfaces:

1. Select the **Interfaces > Ethernet** section of the target system in the tree view of the Security Editor.
2. Right-click on the *wan* object and choose **Properties**, the **Ethernet Properties** dialog box will be displayed.
3. In the **IP Address** control, enter 10.0.0.2
4. Click the **OK**.

1. Select the **Interfaces > Ethernet** section of the target system in the tree view of the Security Editor.
2. Right-click on the *lan* object and choose **Properties**, the **Ethernet Properties** dialog box will be displayed.
3. In the **IP Address** control, enter 10.0.0.2
4. Click the **OK**.

Create an interface group:

1. Select the **Interfaces > Interface Groups** section of the target system in the tree view of the Security Editor.
2. Choose **New Interface Group...** from the context menu, the **Interface Group Properties** dialog box will be displayed.
3. Specify a suitable name for the interface group, for instance TransparentGroup.
4. In the **Interfaces** control, add *lan* and *wan*
5. Click the **OK**.

Add a Switch Route:

1. Select the **Routing > Routes** section of the target system in the tree view of the Security Editor.
2. Choose **New Switch Route...** from the context menu, the **Switch Route Properties** dialog box will be displayed.
3. Specify a suitable name for the route, for instance TransparentGroup.
4. Enter following:
  - **Interface:** Transparentgroup
  - **Network:** 10.0.0.0/24

- **Interface:** Transparentgroup
- **Metric:** 0

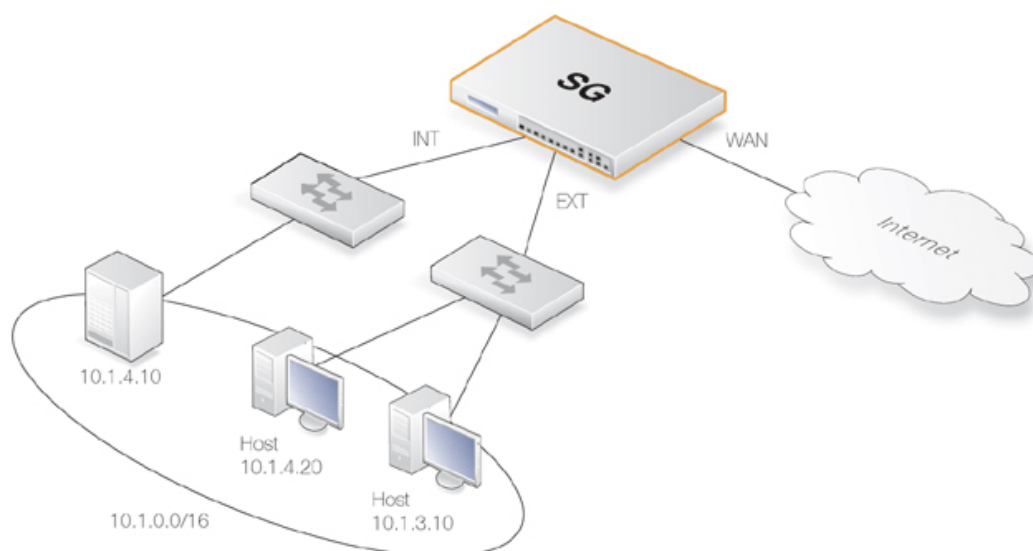
5. Click the **OK**.

Configure the rules:

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance HTTPAllow.
4. Enter following:
  - **Action:** Allow
  - **Source Interface:** lan
  - **Destination Interface:** any
  - **Source Network:** 10.0.0.0/24
  - **Destination Network:** 0.0.0.0/0 (all-nets)
5. Under the **Service** tab, choose *http* in the **Pre-defined** control
6. Click the **OK**.

## Scenario 2

**Figure 4.9. Transparent mode scenario 2**



Here the Clavister Security Gateway in Transparent Mode separates server resources from an internal network by connecting them to a separate interface without the need for different address ranges. All hosts connected to LAN and DMZ (the lan and dmz interfaces) share the 10.0.0.0/24 address space. As this is configured using Transparent Mode any IP address can be used for the servers, and there is no need for the hosts on the internal network to know if a resource is on the same network or placed on the DMZ. The hosts on the internal network are allowed to communicate with an HTTP server on DMZ while the HTTP server on the DMZ can be reached from the internet. The gateway is transparent between the DMZ and LAN while traffic can be subjected to the IP rule-set.

**Example 4.14. Setting up Transparent Mode - Scenario 2**

Configure a **Switch Route** over the LAN and DMZ interfaces for address range 10.0.0.0/24 (assume the WAN interface is already configured).

**Clavister FineTune**

Configure the interfaces:

Similar as shown in the previous example, first, we need to specify the involving interfaces *lan*, and *dmz* using the example IP addresses for this scenario.

Interface Groups:

Similar as shown in the previous example. Configure both interfaces *lan* and *dmz* into the same group.

Switch Route:

Similar as shown in the previous example. Set up the switch route with the new interface group created earlier. Configure the rules:

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance HTTP-LAN-to-DMZ.
4. Enter following:
  - **Action:** Allow
  - **Source Interface:** lan
  - **Destination Interface:** dmz
  - **Source Network:** all-nets
  - **Destination Network:** 10.1.4.10
5. Under the **Service** tab, choose *http* in the **Pre-defined** control
6. Click the **OK**.
7. Select the **Rules** section of the target system in the tree view of the Security Editor.
8. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
9. Specify a suitable name for the rule, for instance HTTP-WAN-to-DMZ.
10. Enter following:
  - **Action:** SAT
  - **Source Interface:** wan
  - **Destination Interface:** dmz
  - **Source Network:** all-nets
  - **Destination Network:** ip\_wan
11. Under the **Service** tab, choose *http* in the **Pre-defined** control
12. Under the **Address Translation** tab, choose *Destination IP Address* and enter *10.1.4.10* in the **New IP Address** control.
13. Click the **OK**.
14. Select the **Rules** section of the target system in the tree view of the Security Editor.
15. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
16. Specify a suitable name for the rule, for instance HTTP-LAN-to-DMZ.
17. Enter following:

- **Action:** Allow
- **Source Interface:** wan
- **Destination Interface:** dmz
- **Source Network:** all-nets
- **Destination Network:** ip\_wan

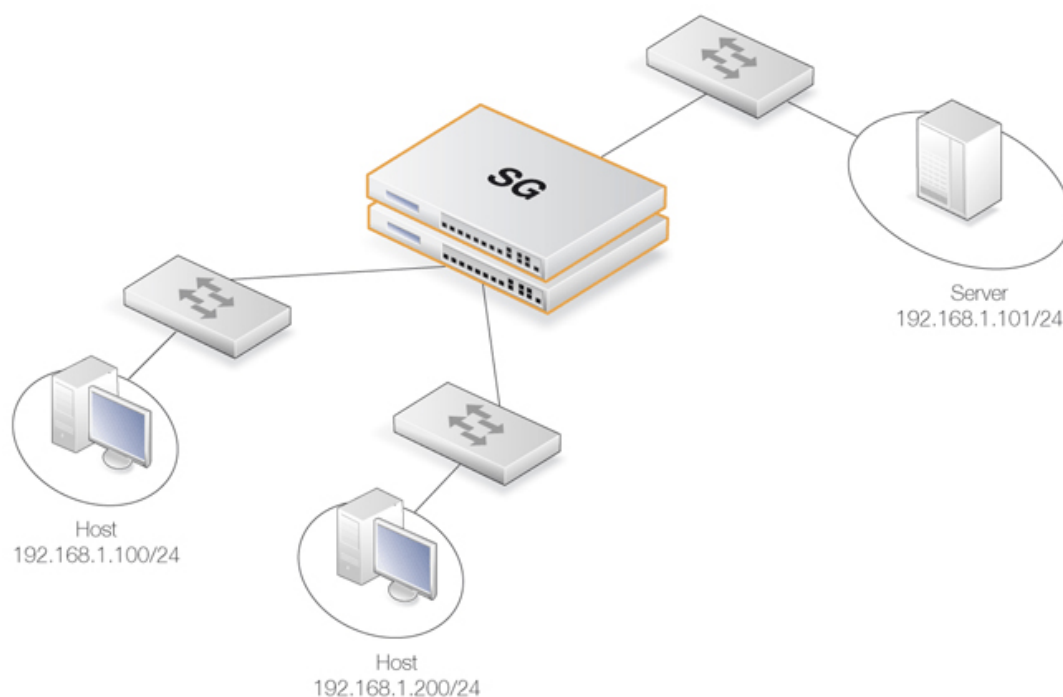
18. Under the **Service** tab, choose *http* in the **Pre-defined** control

19. Click the **OK**.

## 4.7.6. Spanning Tree BPDU Support

CorePlus includes support for relaying the Bridge Protocol Data Units (BPDUs) across the Clavister Security Gateway. BPDU frames carry Spanning Tree Protocol (STP) messages between layer 2 switches in a network. STP allows the switches to understand the network topology and avoid the occurrences of loops in the switching of packets.

**Figure 4.10. An example BPDU relaying scenario**



The scenario above illustrates a situation where BPDU messages would occur if the administrator enables the switches to run the STP protocol. Two Clavister Security Gateways are deployed in transparent mode between the two sides of the network. The switches on either side of the gateway need to communicate and require CorePlus to relay switch BPDU messages in order that packets do not loop between the gateways.

### BPDU Relaying Implementation

The CorePlus BPDU relaying implementation only carries STP messages. These STP messages can be of three types:

- Normal Spanning Tree Protocol (STP)
- Rapid Spanning Tree Protocol (RSTP)
- Multiple Spanning Tree Protocol (MSTP)
- Cisco proprietary PVST+ Protocol (Per VLAN Spanning Tree Plus)

CorePlus checks the contents of BPDU messages to make sure the content type is supported. If it isn't the frame is dropped.

### **Enabling/Disabling BPDU Relaying**

BPDU relaying is disabled by default and can be controlled through the Advanced Setting **RelaySTP** (see Chapter 12, *Advanced Settings*). Logging of BPDU messages can also be controlled through this setting. When enabled, all incoming STP, RSTP and MSTP BPDU messages are relayed to all transparent interfaces in the same routing table, except the incoming interface.



---

# Chapter 5. DHCP Services

This chapter describes DHCP services in CorePlus.

- Overview, page 107
- DHCP Servers, page 108
- DHCP Relaying, page 109

## 5.1. Overview

DHCP (Dynamic Host Configuration Protocol) is a protocol that allows network administrators to automatically assign IP numbers to computers on a network.

The DHCP server implements the task of assigning and managing IP addresses from specified address pools to DHCP clients. When a DHCP server receives a request from a DHCP client, it returns the configuration parameters (such as an IP address, a MAC address, a domain name, and a lease for the IP address) to the client in a unicast message. Because the DHCP server maintains configurations for several subnets, an administrator only needs to update a single, central server.

Compared to the static assignment where the client owns the address, dynamic addressing by the DHCP server leases the address to each client for a pre-defined period of time. During the life cycle of the lease, the client has permission to keep the assigned address and is guaranteed to have no address collision with other clients. Before the expiration of the lease, the client needs to renew the lease from the server, so it can keep using its IP address. The client may also decide at any time that it no longer wishes to use the IP address it was assigned, and may terminate the lease by releasing the IP address. The lease time can be configured in the DHCP server by the administrator.



## 5.2. DHCP Servers

CorePlus includes DHCP server functionality with the ability to act as multiple servers. In other words, a gateway can be used to provision DHCP clients with different address information depending on what network segment they are located on.

A number of standard options can be configured for each DHCP server instance:

- IP Address
- Netmask
- Subnet
- Gateway Address
- Domain Name
- Lease Time
- DNS Servers
- WINS Servers
- Next Server

In addition, *Custom Options* can be specified in order to have the DHCP servers hand out all types of options supported by the DHCP standard.

DHCP servers assign and manage IP addresses from specified address pools within a system. DHCP servers are not limited to serving a single range of IP addresses but can also use single hosts, networks, ranges of IP addresses or a group consisting of these.

### Example 5.1. Setting up a DHCP server

This example shows how to set up a DHCP server which assigns and manages IP addresses from a specific ip address pool. This example presumes you have created an ip range for your DHCP Server.

#### *Clavister FineTune*

1. Select the **Miscellaneous > DHCP Server** section of the target system in the tree view of the Security Editor.
2. Choose **New DHCP Server...** from the context menu, the **DHCP Server Properties** dialog box will be displayed.
3. Specify a suitable name for the server, for instance DHCPServer1.
4. Now enter:
  - **Interface Filter:** lan
  - **IP Address Pool:** DHCPRange1
  - **Netmask:** 255.255.255.0
5. Click **OK**.



#### **Tip**

*DHCP leases are remembered by the system between system restarts.*

## 5.3. DHCP Relaying

With DHCP, clients send requests to locate the DHCP server(s) using broadcast messages. However, broadcasts are normally only propagated across the local network. This means that the DHCP server and client would always need to be in the same physical network area to be able to communicate. In a large Internet-like environment, this means there has to be a different server on every network. This problem is solved by the use of a DHCP relayer.

A DHCP relayer takes the place of the DHCP server in the local network to act as the link between the client and the remote DHCP server. It intercepts requests from clients and relays them to the server. The server then responds to the relay, which forwards the response to the client. The DHCP relayers follow the BOOTP relay agent functionality and retain the BOOTP message format and communication protocol, and hence, they are often called BOOTP relay agents.

### Example 5.2. Setting up a DHCP relayer

This example allows clients on VLAN interfaces to obtain IP addresses from a DHCP server. It's assumed the gateway is configured with VLAN interfaces, "vlan1" and "vlan2", that use DHCP relaying, and the DHCP server IP address is defined in the address book as "ip-dhcp". CorePlus will install a route for the client when it has finalized the DHCP process and obtained an IP.

#### *Clavister FineTune*

Adding VLAN interfaces vlan1 and vlan2 that should relay to an interface group named as ipgrp-dhcp:

1. Select the **Interfaces > Interface Groups** section of the target system in the tree view of the Security Editor.
2. Choose **New Interface Group...** from the context menu, the **DHCP Relay Properties** dialog box will be displayed.
3. Specify a suitable name for the Interface Group, for instance ipgrp-dhcp.
4. In the **Interfaces** control, add "vlan1" and "vlan2" to the list.
5. Click **OK**.

Adding a DHCP relay named as "vlan-to-dhcpserver":

1. Select the **Routing > DHCP Relay** section of the target system in the tree view of the Security Editor.
2. Choose **New DHCP Relay...** from the context menu, the **DHCP Relay Properties** dialog box will be displayed.
3. Specify a suitable name for the Interface Group, for instance vlan-to-dhcpserver.
4. Now enter:
  - **Action:** Relay
  - **Source Interface:** ipgrp-dhcp
  - **DHCP Server to relay to:** ip-dhcp
5. Under the **Add Route** tab, check **Add dynamic routes for this relayed DHCP lease**.
6. Click **OK**.



---

# Chapter 6. Security Mechanisms

This chapter describes CorePlus security features.

- Access Rules, page 111
- Application Layer Gateways, page 114
- Intrusion Detection and Prevention, page 134
- Anti-Virus, page 140
- Web Content Filtering, page 144
- Denial-Of-Service Attacks (DoS), page 158
- Blacklisting Hosts and Networks, page 163

## 6.1. Access Rules

### 6.1.1. Introduction

One of the principal functions of CorePlus is to allow only authorized connections access to protected data resources. Access control is primarily addressed by the CorePlus IP rule-set in which a range of protected LAN addresses are treated as trusted hosts, and traffic flow from untrusted sources is restricted from entering trusted areas.

Before a new connection is checked against the IP rule-set, CorePlus offers the option to check the connection source against a set of *Access Rules*. Access Rules allows the administrator to specify what traffic source is expected on a given interface and also to automatically drop traffic originating from specific sources. AccessRules can provide an efficient and targeted initial filter of new connection attempts.

Specifying Access Rules is not a requirement and many administrators may feel that the checks done against new connections further down the packet processing flow are sufficient. If Access Rule lookup does not produce a match, CorePlus will continue onwards with the normal packet processing flow. AccessRules can however be useful as an initial fast security check of traffic.

A specific application of Access Rules is as a means to combat IP spoofing, an attack technique which is described below.

### 6.1.2. IP spoofing

Traffic that pretends it comes from a trusted host can be sent by an attacker to try and get past a gateway's security mechanisms. Such an attack is commonly known as *Spoofing*.

IP spoofing is one of the most common spoofing attacks. Trusted IP addresses are used to bypass filtering. The header of an IP packet indicating the source address of the packet is modified by the attacker to be a local host address. The gateway will believe the packet came from a trusted source. Although the packet source cannot be responded to correctly, there is the potential for unnecessary network congestion to be created and potentially a Denial of Service (DoS) condition could occur. Even if the gateway is able to detect a DoS condition, it is hard to trace or stop it because of its nature.

VPNs provide one means of avoiding spoofing but where a VPN is not the appropriate solution then Access Rules can provide an anti-spoofing capability by providing an extra filter for source address verification. An Access Rule can verify that packets arriving at a given interface do not have a source address which is associated with a network of another interface. In other words:

- Any incoming traffic with a source IP address belonging to a local trusted host is NOT allowed.
- Any outgoing traffic with a source IP address belonging to an outside untrusted network is NOT allowed.

The first point prevents an outsider from using a local host's address as its source address. The second point prevents any local host from launching the spoof.

## 6.1.3. Access Rule Settings

The configuration of an access rule is similar to other types of rules. It contains **Filtering Fields** as well as the **Action** to take. If there is a match, the rule is triggered, and CorePlus will carry out the specified Action.

### Access Rule Filtering Fields

The Access Rule filtering fields used to trigger a rule are:

- **Interface:** The interface that the packet arrives on.
- **Network:** The IP span that the sender address should belong to.

### Access Rule Action

The Access Rule actions that can be specified are:

- **Drop:** Discard the packets that match the defined fields.
- **Accept:** Accept the packets that match the defined fields for further inspection in the rule-set.
- **Expect:** If the sender address of the packet matches the **Network** specified by this rule, the receiving interface is compared to the specified interface. If the interface matches, the packet is accepted in the same way as an **Accept** action. If the interfaces do not match, the packet is dropped in the same way as a **Drop** action.



#### *Note*

*Logging can be enabled on demand for these Actions.*

### Troubleshooting Access Rule Related Problems

It should be noted that Access Rules are a first filter of traffic before any other CorePlus modules can see it. Sometimes problems can appear, such as setting up VPN tunnels, precisely because of this. It is always advisable to check Access Rules when troubleshooting puzzling problems in case a rule is preventing some other function, such as VPN tunnel establishment, from working properly.

#### **Example 6.1. Setting up an Access Rule**

A rule is to be defined that ensures no traffic with a source address not within the lannet network is received on the lan interface.

**Clavister FineTune**

1. Select the **Miscellaneous > Access** section of the target system in the tree view of the Security Editor.
2. Choose **New Access...** from the context menu. The **Access Properties** dialog box will be displayed.
3. Specify a suitable name for the access rule, for instance lan\_Access.
4. Now enter:
  - **Interface:** lan
  - **Network:** lannet
5. Click **OK**.

## 6.2. Application Layer Gateways

### 6.2.1. Overview

To complement low-level packet filtering, which only inspects packet headers in protocols such as IP, TCP, UDP, and ICMP, Clavister Security Gateways provide *Application Layer Gateways* (ALGs) which provide filtering at the higher application OSI level.

An ALG acts as a mediator in accessing commonly used Internet applications outside the protected network, e.g. Web access, file transfer, and multimedia transfer. ALGs provide higher security than packet filtering since they are capable of scrutinizing all traffic for a specific service protocol and perform checks at the uppermost levels of the TCP/IP stack.

The following protocols are supported by CorePlus ALGs:

- HTTP
- FTP
- SMTP
- H.323

### 6.2.2. Hyper Text Transfer Protocol

Hyper Text Transfer Protocol (HTTP) is the primary protocol used to access the World Wide Web (WWW). It is a connectionless, stateless, application layer protocol based on a request/response architecture. The client, such as a Web browser, sends a request by establishing a TCP/IP connection to a particular port (usually port 80) on a remote server. The server answers with a response string, followed by a message of its own. That message might be, for example, an HTML file to be shown in the Web browser or an ActiveX component to be executed on the client, or an error message.

The HTTP protocol faces particular issues because of the wide variety of web sites that can be accessed and the range of file types that can be downloaded as a result of such access. Two mechanisms that exist in CorePlus to specifically handle this are Web Content Filtering and Anti Virus scanning.

### 6.2.3. File Transfer Protocol

File Transfer Protocol (FTP) is a TCP/IP-based protocol for exchanging files between a client and a server. The client initiates the connection by connecting to the FTP server. Normally the client needs to authenticate itself by providing a predefined login and password. After granting access, the server will provide the client with a file/directory listing from which it can download/upload files (depending on access rights). The FTP ALG is used to manage FTP connections through the Clavister Security Gateway.

#### FTP Connections

FTP uses two communication channels, one for control commands and one for the actual files being transferred. When an FTP session is opened, the FTP client establishes a TCP connection (the control channel) to port 21 (by default) on the FTP server. What happens after this point depends on the mode of FTP being used.

#### Modes

There are two modes, *active* and *passive*, describing the role of server in respect to opening the data channels.

In active mode, the FTP client sends a command to the FTP server indicating what IP address and port the server should connect to. The FTP server establishes the data channel back to the FTP client using the received address information.

In passive mode, the data channel is opened by the FTP client to the FTP server, just like the command channel. This is the often recommended default mode for FTP clients though some advice may recommend the opposite.

## FTP Security Issues

Both modes of FTP operation present problems for gateways. Consider a scenario where an FTP client on the internal network connects through the gateway to an FTP server on the Internet. The IP rule is then configured to allow network traffic from the FTP client to port 21 on the FTP server.

When active mode is used, CorePlus is not aware that the FTP server will establish a new connection back to the FTP client. Therefore, the incoming connection for the data channel will be dropped. As the port number used for the data channel is dynamic, the only way to solve this is to allow traffic from all ports on the FTP server to all ports on the FTP client. Obviously, this is not a good solution.

When passive mode is used, the gateway does not need to allow connections from the FTP server. On the other hand, CorePlus still does not know what port the FTP client tries to use for the data channel. This means that it has to allow traffic from all ports on the FTP client to all ports on the FTP server. Although this is not as insecure as in the active mode case, it still presents a potential security threat. Furthermore, not all FTP clients are capable of using passive mode.

## The Solution

The FTP ALG solves this problem by fully reassembling the TCP stream of the command channel and examining its contents. Thus, the gateway knows what port to be opened for the data channel. Moreover, the FTP ALG also provides functionality to filter out certain control commands and provide a basic buffer overrun protection.

The most important feature of the FTP ALG is its unique capability to perform on-the-fly conversion between active and passive mode. The conversion can be described as follows:

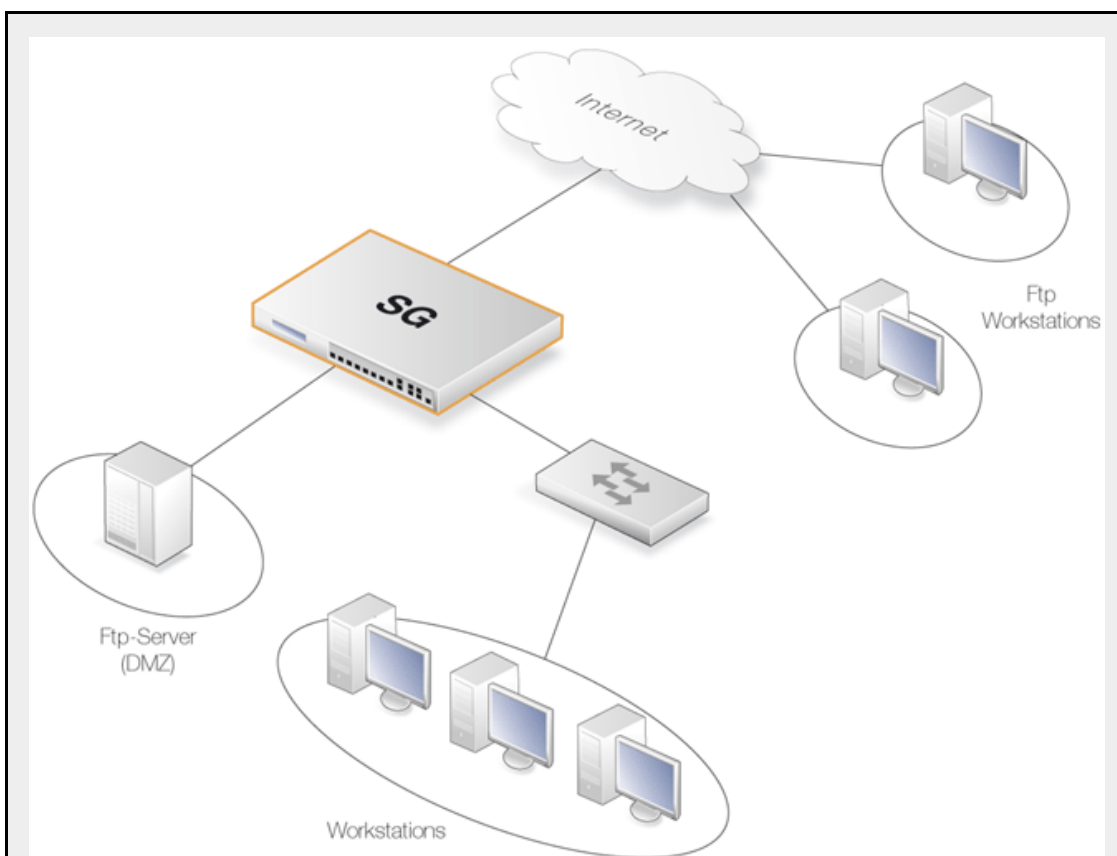
- The FTP client can be configured to use passive mode, which is the recommended mode for clients.
- The FTP server can be configured to use active mode, which is the safer mode for servers.
- When an FTP session is established, the Clavister Security Gateway will automatically and transparently receive the passive data channel from the FTP client and the active data channel from the server, and tie them together.

This implementation results in both the FTP client and the FTP server working in their most secure mode. The conversion also works the other way around, that is, with the FTP client using active mode and the FTP server using passive mode.

### Example 6.2. Protecting an FTP Server with ALG

As shown, an FTP Server is connected to the Clavister Security Gateway on a DMZ with private IP addresses, shown below:





To make it possible to connect to this server from the Internet using the FTP ALG, the FTP ALG and rules should be configured as follows:

#### **Clavister FineTune**

Define the ALG:

1. Select the **Local Objects > Application Layer Gateway** section of the target system in the tree view of the Security Editor.
2. Choose **New Application Gateway Layer...** from the context menu, the **Application Gateway Layer Properties** dialog box will be displayed.
3. Specify a suitable name for the object, for instance ftp-inbound.
4. Choose *ftp* in the **Type** control.
5. Click the **Parameters...** button.
6. Check **Allow client to use active mode**.
7. Uncheck **Allow server to use passive mode**.
8. Click **OK** in both dialogs.

Define the Service:

1. Select the **Local Objects > Services** section of the target system in the tree view of the Security Editor.
2. Choose **New Service...** from the context menu, the **Service Properties** dialog box will be displayed.
3. Specify a suitable name for the service, for instance ftp-inbound.
4. Choose *TCP* in the **Type** control.
5. Choose *ftp-inbound* in the **ALG** control.
6. Under the **TCP/UDP Service**, enter 21 as **Destination Port**.
7. Click **OK**.

Define the Rule - Allow connections to the public IP on port 21 and forward that to the internal FTP server:

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** SAT-ftp-inbound
  - **Action:** SAT
  - **Source Interface:** any
  - **Destination Interface:** core
  - **Source Network:** 0.0.0.0/0 (all-nets)
  - **Destination Network:** wan (assuming the external interface has been defined as this)
4. Under the **Service** tab, choose *ftp-inbound* in the **Pre-defined** control.
5. Under the **Address Translation** tab, choose *Destination IP Address* and select *ftp-inbound* in the **New IP Address** control (assume this internal IP address for FTP server has been defined in the Address Book object).
6. Click the **OK**.

Traffic from the internal interface needs to be NATed:

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** NAT-ftp
  - **Action:** NAT
  - **Source Interface:** dmz
  - **Destination Interface:** core
  - **Source Network:** dmznet
  - **Destination Network:** ip\_wan
4. Under the **Service** tab, choose *ftp-inbound* in the **Pre-defined** control.
5. Under the **Address Translation** tab, choose *Use Interface Address* as action.
6. Click the **OK**.

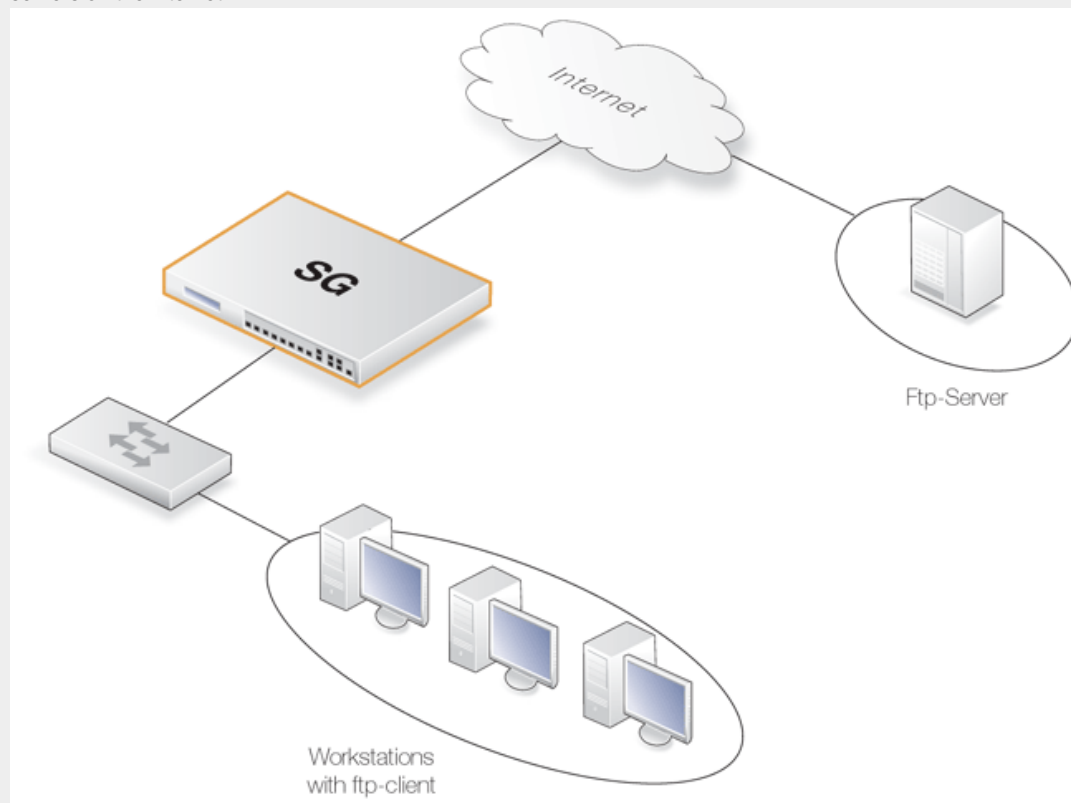
Allow incoming connections (SAT needs a second Allow rule):

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** Allow-ftp
  - **Action:** Allow
  - **Source Interface:** any
  - **Destination Interface:** core
  - **Source Network:** all-nets
  - **Destination Network:** ip\_wan
4. Under the **Service** tab, choose *ftp-inbound* in the **Pre-defined** control.

5. Click the **OK**.

### Example 6.3. Protecting FTP Clients

In this scenario shown below the Clavister Security Gateway is protecting a workstation that will connect to FTP servers on the internet.



To make it possible to connect to these servers from the internal network using the FTP ALG, the FTP ALG and rules should be configured as follows:

#### **Clavister FineTune**

Create the FTP ALG:

1. Select the **Local Objects > Application Layer Gateway** section of the target system in the tree view of the Security Editor.
2. Choose **New Application Gateway Layer...** from the context menu, the **Application Gateway Layer Properties** dialog box will be displayed.
3. Specify a suitable name for the object, for instance ftp-outbound.
4. Choose *ftp* in the **Type** control.
5. Click the **Parameters...** button.
6. Uncheck **Allow client to use active mode**.
7. Check **Allow server to use passive mode**.
8. Click **OK** in both dialogs.

Services:

1. Select the **Local Objects > Services** section of the target system in the tree view of the Security Editor.

2. Choose **New Service...** from the context menu, the **Service Properties** dialog box will be displayed.
3. Specify a suitable name for the service, for instance ftp-outbound.
4. Choose *TCP* in the **Type** control.
5. Choose *ftp-outbound* in the **ALG** control.
6. Under the **TCP/UDP Service**, enter 21 as **Destination Port**.
7. Click **OK**.

Rules (Using Public IPs). The following rule needs to be added to the IP rules if using public IP's; make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules. The service in use is the "ftp-outbound", which should be using the ALG definition "ftp-outbound" as described earlier.

Allow connections to ftp-servers on the outside:

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** Allow-ftp-outbound
  - **Action:** Allow
  - **Source Interface:** lan
  - **Destination Interface:** wan
  - **Source Network:** lannet
  - **Destination Network:** all-nets
4. Under the **Service** tab, choose *ftp-outbound* in the **Pre-defined** control.
5. Click the **OK**.

Rules (Using Private IPs). If the gateway is using private IP's, the following NAT rule need to be added instead:

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** NAT-ftp-outbound
  - **Action:** NAT
  - **Source Interface:** lan
  - **Destination Interface:** wan
  - **Source Network:** lannet
  - **Destination Network:** all-nets
4. Under the **Service** tab, choose *ftp-outbound* in the **Pre-defined** control.
5. Under the **Address Translation** tab, choose *Use Interface Address* as action.
6. Click the **OK**.

## 6.2.4. Simple Mail Transfer Protocol

Simple Mail Transfer Protocol (SMTP) is a text based protocol that is used for transferring email over the internet.

Key features of the SMTP ALG are:

- **Rate Limiting** - A maximum allowable rate of email messages can be specified.
- **Email address blacklisting** - A blacklist of email addresses can be specified so that mail from those addresses is blocked.
- **Email address whitelisting** - A whitelist of email addresses can be specified so that mail from those addresses is allowed to pass by the ALG.
- **MIME Checking** - Mail attachment file content can be checked against its filetype. A list of all filetypes checked can be found in Appendix E, *Anti-Virus MIME filetypes*.
- **Anti-Virus Scanning** - The CorePlus Anti-Virus module can scan email attachments searching for malicious code.

## 6.2.5. H.323

H.323 is a standard approved by the International Telecommunication Union to allow compatibility in video conference transmissions over IP networks. It is used for real-time audio, video and data communication over packet-based networks such as the internet. It specifies the components, protocols and procedures for providing such multimedia communication, including Internet phone and voice-over-IP (VoIP).

### H.323 Components

H.323 consists of four main components:

<b>Terminals</b>	Devices used for audio and optionally video or data communication. eg. phones, conferencing units, or "software phones" such as NetMeeting).
<b>Gateways</b>	An H.323 gateway connects two dissimilar networks and translates traffic between them. It provides connectivity between H.323 networks and non-H.323 networks such as public switched telephone networks (PSTN), translating protocols and converting media them. A gateway is not required for communication between two H.323 terminals.
<b>Gatekeepers</b>	The Gatekeeper is a component in the H.323 system which is used for addressing, authorization and authentication of terminals and gateways. It can also take care of bandwidth management, accounting, billing and charging. The gatekeeper may allow calls to be placed directly between endpoints, or it may route the call signaling through itself to perform functions such as follow-me/find-me, forward on busy, etc. It is needed when there is more than one H.323 terminal behind a NATing device with only one public IP.
<b>Multipoint Control Units</b>	MCUs provide support for conferences of three or more H.323 terminals. All H.323 terminals participating in the conference call have to establish a connection with the MCU. The MCU then manages the calls, resources, video and audio codecs used in the call.

### H.323 Protocols

The different protocols used in implementing H.323 are:

<b>H.225 RAS Signaling and Call Control (Setup) Signaling</b>	Used for call signaling. It used to establish a connection between two H.323 endpoints. This call signal channel is opened between two H.323 endpoints or between a H.323 endpoint and a gatekeeper. For communication between two H.323 endpoints, TCP 1720 is used. When connecting to a gatekeeper, UDP port 1719 (H.225 RAS messages) are used.
<b>H.245 Media Control and Transport</b>	Provides control of multimedia sessions established between two H.323 endpoints. It's most important task is to negotiate opening and closing of logical channels. A logical channel is, for instance, an audio channel used for voice communication. Video and T.120 channels are also called logical channels during negotiation.
<b>T.120</b>	A suite of communication and application protocols. Depending on the type of H.323 product, T.120 protocol can be used for application sharing, file transfer as well as for conferencing features such as whiteboards.

## H.323 ALG features

The H.323 ALG is a flexible application layer gateway that allows H.323 devices such as H.323 phones and applications to make and receive calls between each other when connected via private networks secured by Clavister Security Gateways.

The H.323 specification was not designed to handle NAT, as IP addresses and ports are sent in the payload of H.323 messages. The H.323 ALG modifies and translates H.323 messages to make sure that H.323 messages will be routed to the correct destination and allowed through the Clavister Security Gateway.

The H.323 ALG has the following features:

- The H.323 ALG supports version 5 of the H.323 specification. This specification is built upon H.225.0 v5 and H.245 v10.
- In addition to support voice and video calls, the H.323 ALG supports application sharing over the T.120 protocol. T.120 uses TCP to transport data while voice and video is transported over UDP.
- To support gatekeepers, the ALG monitors RAS traffic between H.323 endpoints and the gatekeeper, in order to correctly configure the Clavister Security Gateway to let calls through.
- NAT and SAT rules are supported, allowing clients and gatekeepers to use private IP addresses on a network behind the Clavister Security Gateway.

## H.323 ALG Configuration

The H.323 ALG can be configured to suit different usage scenarios. It is possible to configure if TCP data channels should be allowed to traverse the Clavister Security Gateway or not. TCP data channels are used by the T.120 protocol for instance. Also, the maximum number of TCP data channels can be limited to a fixed value. The gatekeeper registration lifetime can be controlled by CorePlus in order to force re-registration of clients within a time frame specified by the administrator.

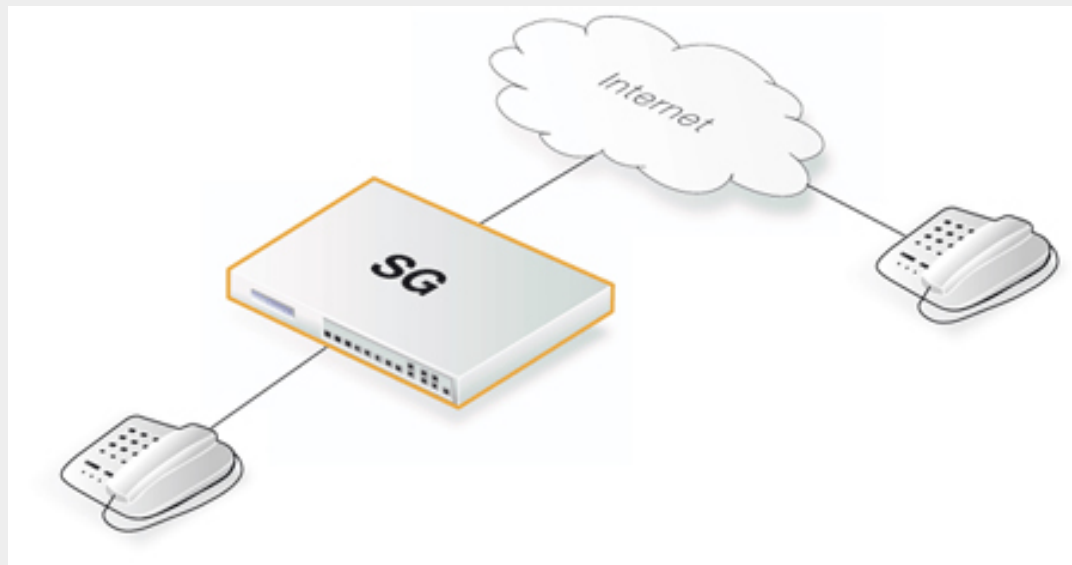
Presented below are some network scenarios where H.323 ALG use is applicable. For each scenario a configuration example of both the ALG and the rules are presented. The three service definitions used in these scenarios are:

- Gatekeeper (UDP ALL > 1719)

- H323 (H.323 ALG, TCP ALL > 1720)
- H323-Gatekeeper (H.323 ALG, UDP > 1719)

#### Example 6.4. Protecting Phones Behind Clavister Security Gateways

In the first scenario a H.323 phone is connected to the Clavister Security Gateway on a network (lannet) with public IP addresses. To make it possible to place a call from this phone to another H.323 phone on the Internet, and to allow H.323 phones on the Internet to call this phone, we need to configure rules. The following rules need to be added to the rule-set, make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules.



#### Clavister FineTune

##### Outgoing Rule:

1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed
3. Enter following:
  - **Name:** H323AllowOut
  - **Action:** Allow
  - **Source Interface:** lan
  - **Destination Interface:** any
  - **Source Network:** lannet
  - **Destination Network:** all-nets
  - **Comment:** Allow outgoing calls
4. Under the **Service** tab, choose *H323* in the **Pre-defined** control
5. Click **OK**

##### Incoming Rule:

1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:

- **Name:** H323AllowIn
  - **Action:** Allow
  - **Source Interface:** any
  - **Destination Interface:** lan
  - **Source Network:** all-nets
  - **Destination Network:** lannet
  - **Comment:** Allow incoming calls
4. Under the **Service** tab, choose *H323* in the **Pre-defined** control
  5. Click **OK**

### Example 6.5. H.323 with private IP addresses

In this scenario a H.323 phone is connected to the Clavister Security Gateway on a network with private IP addresses. To make it possible to place a call from this phone to another H.323 phone on the Internet, and to allow H.323 phones on the Internet to call this phone, we need to configure rules. The following rules need to be added to the rule-set, make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules. As we are using private IPs on the phone incoming traffic need to be SATed as in the example below. The object ip-phone below should be the internal IP of the H.323 phone.

#### *Clavister FineTune*

1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** H323Out
  - **Action:** NAT
  - **Source Interface:** lan
  - **Destination Interface:** any
  - **Source Network:** lannet
  - **Destination Network:** all-nets
  - **Comment:** Allow outgoing calls
4. Under the **Service** tab, choose *H323* in the **Pre-defined** control
5. Click **OK**

Incoming Rule:

1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed
3. Enter following:
  - **Name:** H323In
  - **Action:** SAT
  - **Source Interface:** any
  - **Destination Interface:** core
  - **Source Network:** all-nets

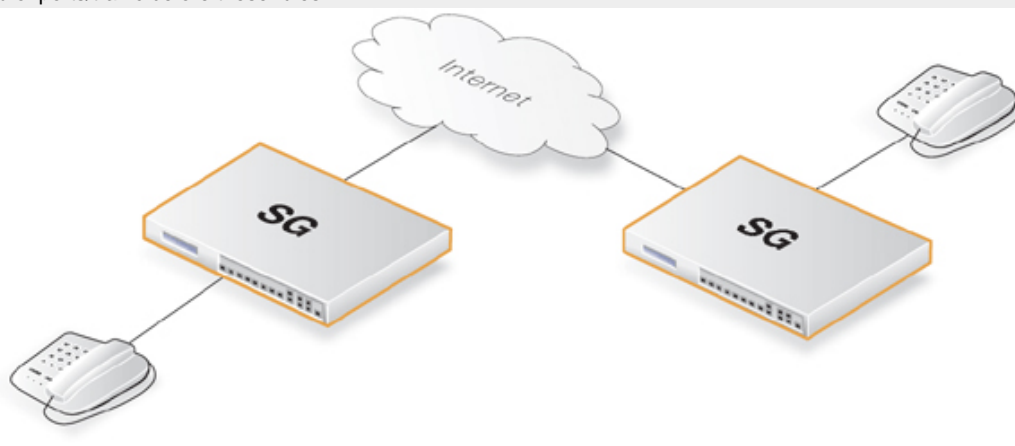


- **Destination Network:** ip\_wan (external IP of the gateway)
4. Under the **Service** tab, choose *H323* in the **Pre-defined** control.
  5. Under the **Address Translation** tab, choose *Destination IP Address* and enter *ip-phone* (IP address of phone) in the **New IP Address** control
  6. Click **OK**
1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed
  3. Enter following:
    - **Name:** H323In
    - **Action:** Allow
    - **Source Interface:** any
    - **Destination Interface:** core
    - **Source Network:** all-nets
    - **Destination Network:** ip\_wan (external IP of the gateway)
    - **Comment:** Allow incoming calls to H.323 phone at ip-phone.
  4. Under the **Service** tab, choose *H323* in the **Pre-defined** control.
  5. Click **OK**

To place a call to the phone behind the Clavister Security Gateway, place a call to the external IP address on the gateway. If multiple H.323 phones are placed behind the gateway, one SAT rule has to be configured for each phone. This means that multiple external addresses have to be used. However, it is preferred to use a H.323 gatekeeper as in the "H.323 with Gatekeeper" scenario, as this only requires one external address.

#### Example 6.6. Two Phones Behind Different Clavister Security Gateways

This scenario consists of two H.323 phones, each one connected behind the Clavister Security Gateway on a network with public IP addresses. In order to place calls on these phones over the Internet, the following rules need to be added to the rule listings in both gateways. Make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules.



**Clavister FineTune**  
Outgoing Rule:

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** H323AllowOut
  - **Action:** Allow
  - **Source Interface:** lan
  - **Destination Interface:** any
  - **Source Network:** lannet
  - **Destination Network:** all-nets
  - **Comment:** Allow outgoing calls
4. Under the **Service** tab, choose *H323* in the **Pre-defined** control.
5. Click the **OK**.

Incoming Rule:

1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** H323AllowIn
  - **Action:** Allow
  - **Source Interface:** any
  - **Destination Interface:** lan
  - **Source Network:** all-nets
  - **Destination Network:** lannet
  - **Comment:** Allow incoming calls
4. Under the **Service** tab, choose *H323* in the **Pre-defined** control.
5. Click **OK**

### Example 6.7. Using Private IP Addresses

This scenario consists of two H.323 phones, each one connected behind the Clavister Security Gateway on a network with private IP addresses. In order to place calls on these phones over the Internet, the following rules need to be added to the rule-set in the gateway, make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules. As we are using private IPs on the phones, incoming traffic need to be SATed as in the example below. The object ip-phone below should be the internal IP of the H.323 phone behind each gateway.

#### *Clavister FineTune*

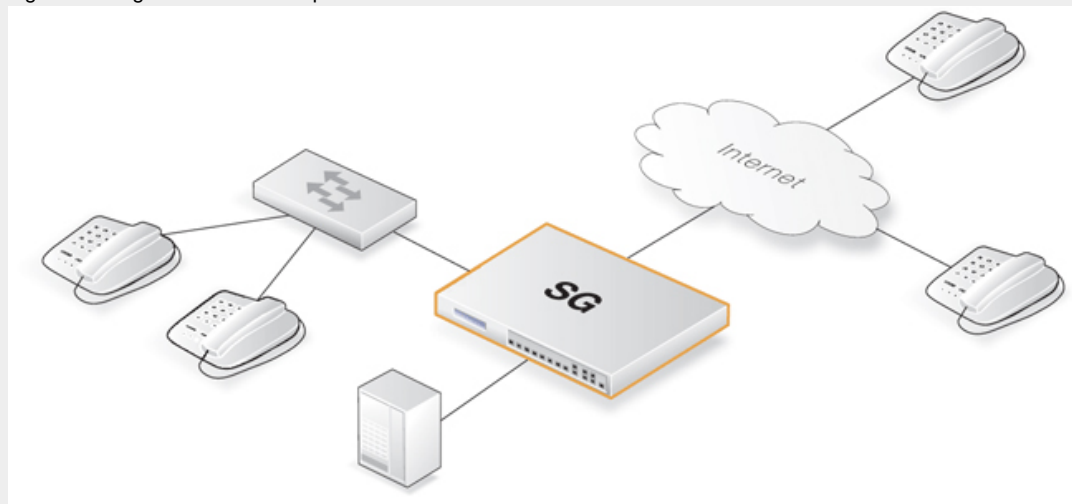
1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed
3. Enter following:
  - **Name:** H323Out
  - **Action:** NAT

- **Source Interface:** lan
  - **Destination Interface:** any
  - **Source Network:** lannet
  - **Destination Network:** all-nets
  - **Comment:** Allow outgoing calls
4. Under the **Service** tab, choose *H323* in the **Pre-defined** control.
  5. Click **OK**
- Incoming Rule:
1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** H323In
    - **Action:** SAT
    - **Source Interface:** any
    - **Destination Interface:** core
    - **Source Network:** all-nets
    - **Destination Network:** ip\_wan (external IP of the gateway)
  4. Under the **Service** tab, choose *H323* in the **Pre-defined** control
  5. Under the **Address Translation** tab, choose *Destination IP Address* and enter *ip-phone* (IP address of phone) in the **New IP Address** control
  6. Click **OK**
1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** H323In
    - **Action:** Allow
    - **Source Interface:** any
    - **Destination Interface:** core
    - **Source Network:** all-nets
    - **Destination Network:** ip\_wan (external IP of the gateway)
    - **Comment:** Allow incoming calls to H.323 phone at ip-phone.
  4. Under the **Service** tab, choose *H323* in the **Pre-defined** control.
  5. Click **OK**

To place a call to the phone behind the Clavister Security Gateway, place a call to the external IP address on the gateway. If multiple H.323 phones are placed behind the gateway, one SAT rule has to be configured for each phone. This means that multiple external addresses have to be used. However, it is preferable to use an H.323 gatekeeper as this only requires one external address.

### Example 6.8. H.323 with Gatekeeper

In this scenario, a H.323 gatekeeper is placed in the DMZ of the Clavister Security Gateway. A rule is configured in the gateway to allow traffic between the private network where the H.323 phones are connected on the internal network and to the Gatekeeper on the DMZ. The Gatekeeper on the DMZ is configured with a private address. The following rules need to be added to the rule listings in both gateways, make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules.



#### Clavister FineTune

Incoming Gatekeeper Rules:

1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Action:** H323In
    - **Action:** SAT
    - **Source Interface:** any
    - **Destination Interface:** core
    - **Source Network:** 0.0.0.0/0 (all-nets)
    - **Destination Network:** ip\_wan (external IP of the gateway)
    - **Comment:** SAT rule for incoming communication with the Gatekeeper located at ip-gatekeeper
  4. Under the **Service** tab, choose *H323-Gatekeeper* in the **Pre-defined** control.
  5. Under the **Address Translation** tab, choose *Destination IP Address* and enter ip-gatekeeper (IP address of gatekeeper).
  6. Click **OK**
- 
1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** H323In
    - **Action:** Allow
    - **Source Interface:** any
    - **Destination Interface:** core

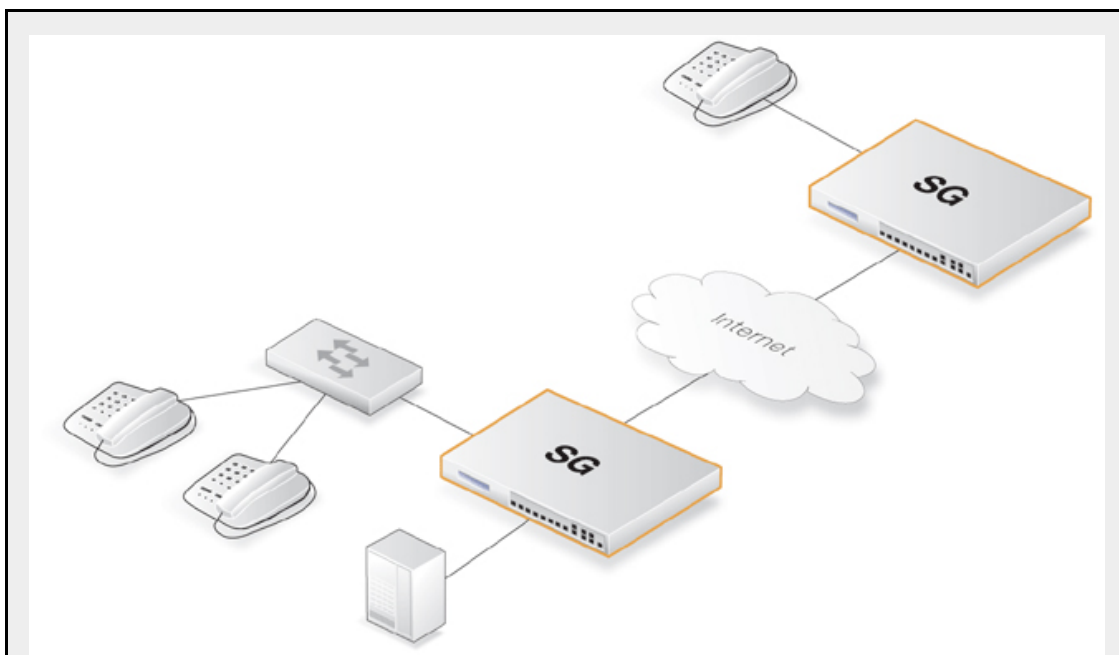
- **Source Network:** all-nets
  - **Destination Network:** ip\_wan (external IP of the gateway)
  - **Comment:** Allow incoming communication with the Gatekeeper
4. Under the **Service** tab, choose *H323-Gatekeeper* in the **Pre-defined** control
  5. Click **OK**
- 
1. Select the **Rules** section of the target system in the tree view of the Security Editor.
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** H323In
    - **Action:** Allow
    - **Source Interface:** lan
    - **Destination Interface:** dmz
    - **Source Network:** lannet
    - **Destination Network:** ip-gatekeeper (IP address of the gatekeeper)
    - **Comment:** Allow incoming communication with the Gatekeeper
  4. Under the **Service** tab, choose *Gatekeeper* in the **Pre-defined** control.
  5. Click the **OK**.

**Note**

*There is no need to specify a specific rule for outgoing calls. CorePlus monitors the communication between "external" phones and the Gatekeeper to make sure that it is possible for internal phones to call the external phones that are registered with the gatekeeper.*

**Example 6.9. H.323 with Gatekeeper and two Clavister Security Gateways**

This scenario is quite similar to scenario 3, with the difference that the Clavister Security Gateway is protecting the "external" phones. The Clavister Security Gateway with the Gatekeeper connected to the DMZ should be configured exactly as in scenario 3. The other Clavister Security Gateway should be configured as below. The rules need to be added to the rule listings, and it should be make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules.



#### Clavister FineTune

1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** H323Out
  - **Action:** NAT
  - **Source Interface:** lan
  - **Destination Interface:** any
  - **Source Network:** lannet
  - **Destination Network:** all-nets
  - **Comment:** Allow outgoing communication with a gatekeeper
4. Under the **Service** tab, choose *H323-Gatekeeper* in the **Pre-defined** control.
5. Click the **OK**

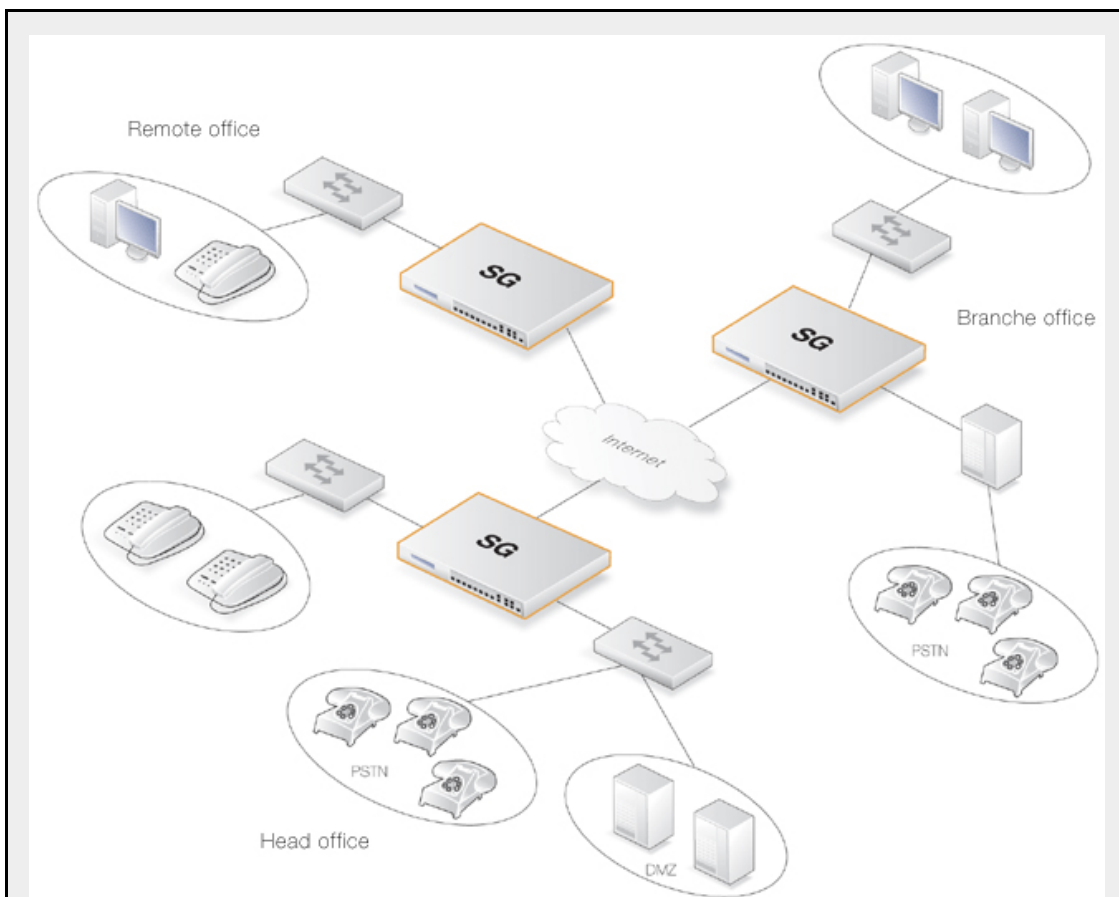


#### Note

*There is no need to specify a specific rule for outgoing calls. CorePlus monitors the communication between "external" phones and the Gatekeeper to make sure that it is possible for internal phones to call the external phones that are registered with the gatekeeper.*

#### Example 6.10. Using the H.323 ALG in a Corporate Environment

This scenario is an example of a more complex network that shows how the H.323 ALG can be deployed in a corporate environment. At the head office DMZ a H.323 Gatekeeper is placed that can handle all H.323 clients in the head-, branch- and remote offices. This will allow the whole corporation to use the network for both voice communication and application sharing. It is assumed that the VPN tunnels are correctly configured and that all offices use private IP-ranges on their local networks. All outside calls are done over the existing telephone network using the gateway (ip-gateway) connected to the ordinary telephone network.



The head office has placed a H.323 Gatekeeper in the DMZ of the corporate Clavister Security Gateway. This gateway should be configured as follows:

#### **Clavister FineTune**

1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** LanToGK
    - **Action:** Allow
    - **Source Interface:** lan
    - **Destination Interface:** dmz
    - **Source Network:** lannet
    - **Destination Network:** ip-gatekeeper (IP address of the gatekeeper)
    - **Comment:** Allow H323. entities on lannet to connect to the Gatekeeper
  4. Under the **Service** tab, choose *Gatekeeper* in the **Pre-defined** control.
  5. Click **OK**
- 
1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** LanToGK

- **Action:** Allow
  - **Source Interface:** lan
  - **Destination Interface:** dmz
  - **Source Network:** lannet
  - **Destination Network:** ip-gatekeeper (IP address of the gatekeeper)
  - **Comment:** Allow H323. entities on lannet to call phones connected to the H.323 Gateway on the DMZ
4. Under the **Service** tab, choose *H323* in the **Pre-defined** control.
  5. Click **OK**
- 
1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** GWTOLan
    - **Action:** Allow
    - **Source Interface:** dmz
    - **Destination Interface:** lan
    - **Source Network:** ip-gateway (IP address of the gateway)
    - **Destination Network:** lannet
    - **Comment:** Allow communication from the Gateway to H.323 phones on lannet
  4. Under the **Service** tab, choose *H323* in the **Pre-defined** control.
  5. Click **OK**
- 
1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** BranchToGW
    - **Action:** Allow
    - **Source Interface:** vpn-branch
    - **Destination Interface:** dmz
    - **Source Network:** branch-net
    - **Destination Network:** ip-gatekeeper,ip-gateway
    - **Comment:** Allow communication with the Gatekeeper on DMZ from the Branch network
  4. Under the **Service** tab, choose *H323-Gatekeeper* in the **Pre-defined** control.
  5. Click **OK**.
- 
1. Go to **Rules**
  2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
  3. Enter following:
    - **Name:** BranchToGW



- **Action:** Allow
  - **Source Interface:** vpn-remote
  - **Destination Interface:** dmz
  - **Source Network:** remote-net
  - **Destination Network:** ip-gatekeeper
  - **Comment:** Allow communication with the Gatekeeper on DMZ from the Remote network
4. Under the **Service** tab, choose *H323-Gatekeeper* in the **Pre-defined** control.
  5. Click **OK**.

### Example 6.11. Configuring remote offices for H.323

If the branch and remote office H.323 phones and applications are to be configured to use the H.323 Gatekeeper at the head office, the Clavister Security Gateways in the remote and branch offices should be configured as follows: (this rule should be in both the Branch and Remote Office gateways).

#### *Clavister FineTune*

1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** ToGK
  - **Action:** Allow
  - **Source Interface:** lan
  - **Destination Interface:** vpn-hq
  - **Source Network:** lannet
  - **Destination Network:** hq-net
  - **Comment:** Allow communication with the Gatekeeper connected to the Head Office DMZ
4. Under the **Service** tab, choose *H323-Gatekeeper* in the **Pre-defined** control.
5. Click **OK**

### Example 6.12. Allowing the H.323 Gateway to register with the Gatekeeper

The branch office Clavister Security Gateway has a H.323 Gateway connected to its DMZ. In order to allow the Gateway to register with the H.323 Gatekeeper at the Head Office, the following rule has to be configured:

#### *Clavister FineTune*

1. Go to **Rules**
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Enter following:
  - **Name:** GWToGK
  - **Action:** Allow

- **Source Interface:** dmz
  - **Destination Interface:** vpn-hq
  - **Source Network:** ip-branchgw
  - **Destination Network:** hq-net
  - **Comment:** Allow the Gateway to communicate with the Gatekeeper connected to the Head Office
4. Under the **Service** tab, choose *H323-Gatekeeper* in the **Pre-defined** control.
  5. Click **OK**

**Note**

*There is no need to specify a specific rule for outgoing calls. CorePlus monitors the communication between "external" phones and the Gatekeeper to make sure that it's possible for internal phones to call the external phones that are registered with the gatekeeper.*

## 6.3. Intrusion Detection and Prevention

### 6.3.1. Overview

#### Intrusion Definition

Computer servers can sometimes have vulnerabilities which leave them exposed to attacks carried by network traffic. Worms, trojans and backdoor exploits are examples of such attacks which, if successful, can potentially compromise or take control of a server. A generic term that can be used to describe these server orientated threats are *intrusions*.

#### Intrusion Detection

Intrusions differ from viruses in that a virus is normally contained in a single file download and this is normally downloaded to a client processor. An intrusion manifests itself as a malicious pattern of internet data aimed at bypassing server security mechanisms. Intrusions are not uncommon and they can constantly evolve as their creation can be automated by the attacker. CorePlus IDP provides an important line of defense against these threats.

Intrusion Detection and Prevention (IDP) is a CorePlus module that is designed to protect against these intrusion attempts. It operates by monitoring network traffic as it passes through the Clavister Security Gateway, searching for patterns that indicate an intrusion is being attempted. Once detected, CorePlus IDP allows steps to be taken to neutralize both the intrusion attempt as well as its source.

#### IDP Issues

In order to have an effective and reliable IDP system, the following issues have to be addressed:

1. What kinds of traffic should be analyzed?
2. What should we searched for in that traffic?
3. What action should be carried out when an intrusion is detected?

#### CorePlus IDP Components

CorePlus IDP addresses the above IDP issues with the following mechanisms:

1. **IDP Rules** are defined up by the administrator to determine what traffic should be scanned.
2. **Pattern Matching** is applied by CorePlus IDP to the traffic that matches an IDP Rule as it streams through the gateway.
3. If CorePlus IDP detects an intrusion then the **Action** specified for the triggering IDP Rule is taken.

IDP Rules, Pattern Matching and IDP Rule Actions are described in the sections which follow.

### 6.3.2. IDP Rules

#### Rule Components

An **IDP Rule** defines what kind of traffic, or service, should be analyzed. An IDP Rule is similar in makeup to an IP Rule. An IDP Rule specifies a given combination source/destination interfaces/addresses as well as being associated with a Service object which defines which protocols to scan. A time schedule can also be associated with an IDP Rule. Most importantly, an IDP Rule specifies the **Action** to take on detecting an intrusion in the traffic targeted by the rule.

### Initial Packet Processing

The initial order of packet processing with IDP is as follows:

1. A packet arrives at the gateway and CorePlus performs normal verification. If the packet is part of a new connection then it is checked against the IP Rule-set before being passed to the IDP module. If the packet is part of an existing connection it is passed straight to the IDP system. If the packet is not part of an existing connection or is rejected by the IP rule-set then it is dropped.
2. The source and destination information of the packet is compared to the set of IDP Rules defined by the administrator. If a match is found, it is passed on to the next level of IDP processing which is pattern matching, described in step below. If there is no match against an IDP rule then the packet is accepted and the IDP system takes no further actions although further actions defined in the IP rule-set are applied eg. address translation, logging.

## 6.3.3. IDP Pattern Matching

### Signatures

In order for IDP to correctly identify an attack, it uses a profile of indicators, or *pattern*, associated with different types of attack. These pre-defined patterns, also known as *signatures*, are stored in a local CorePlus database and are used by the IDP module to analyze traffic for attack patterns. Each IDP signature is designated by a unique number.

Consider the following simple attack example involving an exchange with an FTP server. A rogue user might try to retrieve the password file "passwd" from an FTP server using the FTP command **RETR passwd**. A signature looking for the ASCII text strings *RETR* and *passwd* would find a match in this case, indicating a possible attack. In this example, the pattern is found in plaintext but pattern matching is done in the same way on pure binary data.

### Recognising Unknown Threats

Attackers who build new intrusions often re-use older code. This means their new attacks can appear "in the wild" quickly. To counter this, Clavister IDP uses an approach where the module scans for these reusable components, with pattern matching looking for building blocks rather than the entire complete code patterns. This means that "known" threats as well as new, recently released, "unknown" threats, built with re-used software components, can be protected against.

### Signature Advisories

An *advisory* is a explanatory textual description of a signature. Reading a signature's advisory will explain to the administrator what the signature will search for. Due to the changing nature of the signature database, advisories are not included in Clavister documentation but instead, are available on the Clavister website at:

<http://www.clavister.com/securityportal/advisories/>

### IDP Signature types

IDP offers two signature types which offer differing levels of certainty with regard to threats:

- **Intrusion Protection Signatures (IPS)** - are highly accurate and a match is almost certainly an indicator of a threat. Using the **Protect** action is recommended. These signatures can detect administrative actions and security scanners.
- **Intrusion Detection Signatures (IDS)** - can detect events that may be intrusions- They have lower accuracy than IPS and may give some false positives so that's recommended that the **Audit** action is initially used before deciding to use **Protect**.
- **Policy Signatures** - detect different types of application traffic. They can be used to block certain applications such as filesharing applications and instant messaging.

## 6.3.4. IDP Signature Groups

### Using Groups

Usually, several lines of attacks exist for a specific protocol, and it is best to search for all of them at the same time when analyzing network traffic. To do this, signatures related to a particular protocol are grouped together. For example, all signatures that refer to the FTP protocol form a group. It is best to specify a group that relates to the traffic being searched than be concerned about individual signatures. For performance purposes, the aim should be to have CorePlus search data using the least possible number of signatures.

#### 1. Signature Group Type

The group type is one of the values *IDS*, *IPS* or *Policy*. These types are explained above.

#### 2. Signature Group Category

This second level of naming describes the type of application or protocol. Examples are:

- BACKUP
- DB
- DNS
- FTP
- HTTP

#### 3. Signature Group Sub-Category

The third level of naming further specifies the target of the group and often specifies the application eg. *MSSQL*. The Sub-Category may not be necessary if the *Type* and *Category* are sufficient to specify the group eg. *APP\_ITUNES*.

### Selecting Groups With Clavister FineTune

Clavister FineTune allows selection of signature groups through a hierarchical tree view of all signatures. The *Type is selected*, then the *Category* followed by the *Sub-Category*. Going down further into the tree-view will show the individual signatures in a group. These individual signatures can be turned off or on if desired so that a signature group can be changed to contain only those signatures

which are of interest.

### Listing of IDP Groups

A listing of IDP groupings can be found in Appendix D, *IDP Signature Groups*. The listing shows groups names consisting of the *Category* followed by the *Sub-Category* since the *Type* could be any of IDS, IPS or POLICY.

### Processing multiple actions

For any IDP rule, it's possible to specify multiple actions and an action type such as **Protect** can be repeated. Each action will then have one or more signatures or groups associated with it. When signature matching occurs it is done in a top-down fashion, with matching for the signatures for the first action specified being done first.



#### **Caution against using too many IDP signatures**

*Do not use the entire signature database and avoid using signatures and signature groups unnecessarily. Instead, use only those signatures or groups applicable to the type of traffic you are trying to protect.*

*IDP traffic scanning creates an additional processor load that in most cases shouldn't noticeably degrade performance. Using too many signatures during scanning can make the load on the gateway hardware unnecessarily high, adversely effecting throughput.*

## 6.3.5. IDP Actions

### Action Options

After pattern matching recognises an intrusion in traffic subject to an IDP Rule, the Action associated with that Rule is taken. The administrator can associate one of three Action options with an IDP Rule:

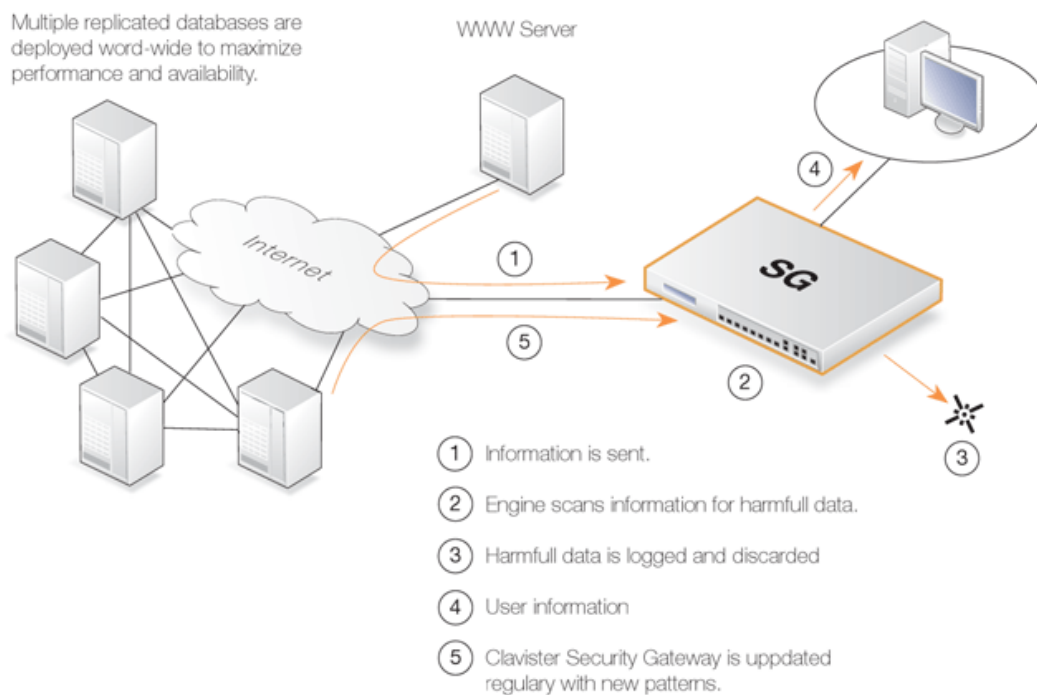
- **Ignore** - Do nothing if an intrusion is detected and allow the connection to stay open
- **Audit** - Allow the connection to stay open but log the event
- **Protect** - This option drops the connection and logs the event (with the additional option to blacklist the source of the connection as described below).

### IDP Blacklisting

The **Protect** option includes the option that the particular host or network that triggers the IDP Rule can be added to a *Blacklist* of offending traffic sources. This means that all subsequent traffic coming from a blacklisted source will be automatically dropped by CorePlus. For more details of how blacklisting functions see Section 6.7, "Blacklisting Hosts and Networks".

## 6.3.6. Subscribing to Clavister IDP

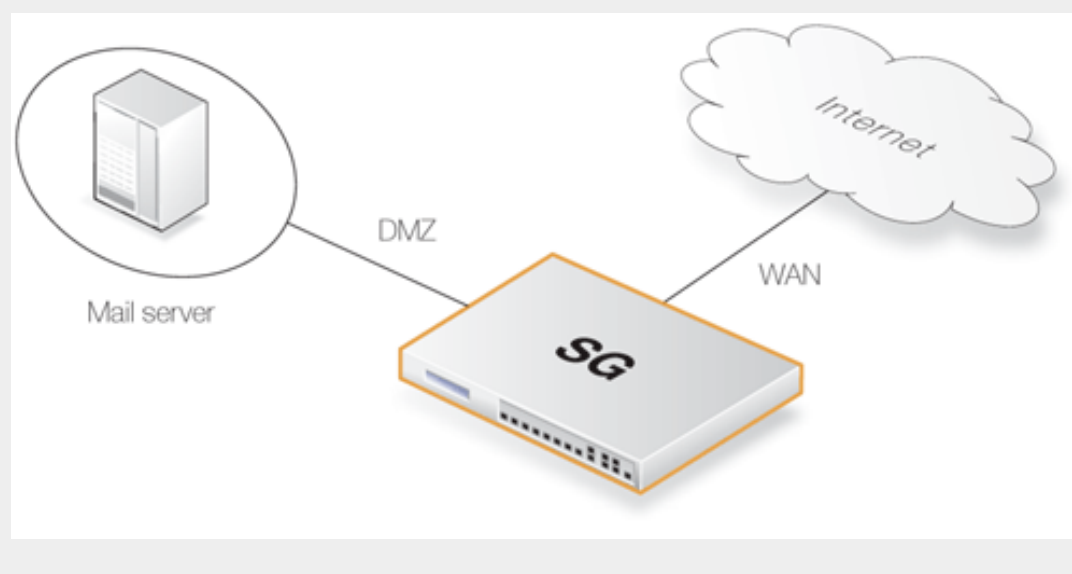
Clavister IDP is purchased as an additional component to the base CorePlus license. It is a subscription service and the subscription means that the IDP signature database can be downloaded to a CorePlus installation and also that the database is regularly updated with the latest intrusion threats. For full details about obtaining the IDP service please refer to Appendix A, *Subscribing to Security Updates*.

**Figure 6.1. IDP Database Updating**

New attacks can be discovered on a daily basis, so it is important to have an up to date signature database in order to protect the network from the latest threats. A new, updated signature database is downloaded automatically by CorePlus system at a configurable interval. This is done via an HTTP connection to the Clavister server network which delivers the latest signature database updates. If the server's signature database has a newer version than the current local database, the new database will be downloaded, replacing the older version.

#### Example 6.13. Setting up IDP for a Mail Server

The following example details the steps needed to set up IDP for a simple scenario where a mail server is exposed to the Internet on the **DMZ** network with a public IP address. The public Internet can be reached through the gateway on the **WAN** interface as illustrated below.



**Clavister FineTune**

Create IDP Rule:

An IDP rule called IDPMailSrvRule will be created, and the *Service* to use is the SMTP service. Source Interface and Source Network defines where traffic is coming from, in this example the external network. The *Destination Interface* and *Destination Network* define where traffic is directed to, in this case the mail server. *Destination Network* should therefore be set to the object defining the mail server.

1. Go to **Intrusion Prevention > Intrusion Prevention Rules**
2. In the context menu select **New Intrusion Prevention Rule...** and the **Intrusion Prevention Rule Properties** dialog will appear.
3. Now enter:
  - **Name:** *IDPMailSrvRule*
  - **Source Interface:** *wan*
  - **Source Network:** *wannet*
  - **Destination Interface:** *dmz*
  - **Destination Network:** *ip\_mailserver*
4. Select the **Service** tab and select the *smtp* service so this IDP rule will be used only on this service
5. Select the **Rule Actions** tab and select the IDP *SMTP\_Server* signature group and the **Protect** option for this action.

Note that it is important to select the **minimum number of IDP signature groups that are applicable to the traffic that is to be scanned** to ensure reasonable performance throughput. Selecting too many groups can degrade throughput.

To summarize, the following will occur: If traffic from the external network, to the mail server is discovered, the IDP will scan the packets. In case the traffic matches any of the signatures in the *IPS\_MAIL\_SMTP* signature group, the connection will be dropped, thus protecting the mail server. If the *Blacklist had been selected instead then the source IP or network would be blacklisted and all future traffic from there blocked.*



## 6.4. Anti-Virus

### 6.4.1. Overview

The CorePlus Anti-Virus module protects against malicious code carried in file downloads. Files may be downloaded as part of a web-page in an HTTP transfer, in an FTP download, or perhaps as an attachment to an email delivered through SMTP. Malicious code in such downloads can have different intents ranging from programs that merely cause annoyance to more sinister aims such as sending back passwords, credit card numbers and other sensitive information. The term "Virus" can be used as a generic description for all forms of malicious code carried in files.

#### Combining with Client Anti-Virus Scanning

Unlike IDP, which is primarily directed at attacks against servers, Anti-Virus scanning is focussed on downloads by clients. CorePlus Anti-Virus scanning is designed to be a compliment to the standard virus scanning normally carried out locally by specialised software installed on client computers. IDP is not intended as a complete substitute for local scanning but rather as an extra shield to boost client protection. Most importantly, it can act as a backup for when local client Anti-Virus scanning is, for some reason, not able to function.

### 6.4.2. Implementation

#### Streaming

As a file transfer is streamed through a Clavister Security Gateway, CorePlus will scan the data stream for the presence of viruses if the Anti-Virus module is enabled. Since files are being streamed and not being read completely into memory, a minimum amount of gateway memory is required and there is minimal effect on overall throughput.

#### Pattern Matching

The inspection process is based on *pattern matching* against a database of known virus patterns and can determine, with a high degree of certainty, if a virus is in the process of being downloaded to a user behind a Clavister Security Gateway. Once a virus is recognized in the contents of a file, the download can be terminated before it completes.

#### Types of Files Scanned

The CorePlus Anti-Virus module is able to scan the following types of downloads:

- HTTP, FTP or SMTP file downloads
- Any uncompressed file type transferred through these protocols
- If the download has been compressed, ZIP and GZIP files can be scanned

The administrator has the option to always drop specific files as well as the option to specify a size limit on scanned files. If no size limit is specified then there is no default upper limit on file sizes.

#### Simultaneous Scans

There is no fixed limit on how many Anti-Virus scans can take place simultaneously in a single Clavister Security Gateway. However the available free memory can place a limit on the number of concurrent scans that can be initiated. The administrator can increase the default amount of free

memory available to Anti-Virus scanning through changing the **AVSE\_MAXMEMORY** advanced setting. This setting specifies what percentage of total memory is to be used for Anti-Virus scanning.

### Protocol Specific Behaviour

Since Anti-Virus scanning is implemented through an Application Level Gateway (ALG), specific protocol specific features are implemented in CorePlus. With FTP, for example, scanning is aware of the dual control and data transfer channels that are opened and can send a request via the control connection to stop a download if a virus in the download is detected.

## 6.4.3. Activation

### Association with an ALG

Activation of Anti-Virus scanning is achieved through an Application Layer Gateway (ALG) associated with the targeted protocol. For instance, an HTTP ALG with Anti-Virus enabled can be created for scanning HTTP downloads. The ALG must then be associated with a given Service which in turn is used by a particular rule defined in the IP Rule-set.

### Creating Anti-Virus Policies

Since IP Rule-set rules are the means by which the Anti-Virus feature is deployed, the deployment can be *policy based*. IP rules can specify that the ALG and it's associated Anti-Virus scanning can apply to traffic going in a given direction and between specific source and destination IP addresses and/or networks. Scheduling can also be applied to virus scanning so that it takes place only at specific times.

## 6.4.4. The Signature Database

### SafeStream

CorePlus Anti-Virus scanning is implemented by Clavister using the "SafeStream" virus signature database. The SafeStream database is created and maintained by Kaspersky, a company which is a world leader in the field of virus detection. The database provides protection against virtually all known virus threats including trojans, worms, backdoor exploits and others. The database is also thoroughly tested to provide near zero false positives.

### Database Updates

The SafeStream database is updated on a daily basis with new virus signatures. Older signatures are seldom retired but instead are replaced with more generic signatures covering several viruses. The local CorePlus copy of the SafeStream database should therefore be updated regularly and this updating service is enabled as part of the subscription to the Clavister Anti-Virus subscription.

## 6.4.5. Subscribing to the Clavister Anti-Virus Service

The Clavister Anti-Virus feature is purchased as an additional component to the base Clavister license and is bought in the form of a renewable subscription. An Anti-Virus subscription includes regular updates of the Kaspersky SafeStream database during the subscription period with the signatures of the latest virus threats.

To subscribe to the Anti-Virus service please refer to the details described in Appendix A, *Subscribing to Security Updates*.

## 6.4.6. Anti-Virus Options

When configuring Anti-Virus scanning in an ALG, the following parameters can be set:

### 1. General options

<b>Mode</b>	When <b>Enabled</b> Anti-Virus is active
<b>Verify MIME type</b>	<p>The MIME type identifies a file's type. For instance a file might be identified as being of type <i>.gif</i> and therefore should contain image data of that type. Some viruses can try to hide inside files by using a misleading file type. A file might pretend to be a <i>.gif</i> file but the file's data will not match that type's data pattern because it is infected with a virus.</p> <p>CorePlus can check that the file's contents matches the MIME type it claims to be. Enabling of this function is recommended to make sure this form of attack cannot allow a virus to get through. The possible MIME types that can be checked are listed in Appendix E, <i>Anti-Virus MIME filetypes</i>.</p>
<b>Max download size</b>	The size of any single component in an single transfer can be limited.
<b>Fail mode behaviour</b>	If a virus scan fails for any reason then the transfer can be dropped or allowed, with the event being logged.

### 2. File type blocking/allowing

**Action** When a particular download file type is encountered, the administrator can explicitly state if the file is to be allowed or blocked as a download.

**File types** The file type to be blocked or allowed eg. GIF, can be added into the list

If a filetype is on the allowed list then it should be noted that MIME matching will still take place even if MIME matching is switched off (providing the filetype is part of the list in Appendix E, *Anti-Virus MIME filetypes*). This is done to guard against an attack that tries to exploit the fact the filetype is on the allowed list.

### 3. Scan exclude option

Certain filetypes may be explicitly excluded from virus-scanning if that is desirable. This can increase overall throughput if an excluded filetype is a type which is commonly encountered in a particular scenario.

#### Example 6.14. Enabling Anti-Virus Scanning

This example shows how to setup an Anti-Virus scanning policy for HTTP traffic from **lannet** to **all-nets** We will assume there is already a NAT rule defined in the IP Rule-set to handle this traffic.

##### *Clavister FineTune*

A. Create a new HTTP Application Layer Gateway (ALG) Object:

1. Go to **Local Objects > Application Layer Gateways**

2. Choose **New Application Layer Gateway...** from the context menu and the **Application Layer Gateway Properties** dialog box will be displayed
3. Specify a name for the ALG, for instance *anti\_virus*
4. Select **HTTP** in the **Type** dropdown list
5. Click the **Parameters...** button and the **HTTP Application Layer Gateway** dialog box will be shown
6. Click the **Antivirus** tab
7. Select **Enabled** in the **Mode** dropdown list
8. Click **OK** for both of the open dialog boxes

B. Next, create a Service object using the new HTTP ALG:

1. Go to **Local Objects > Services**
2. Choose **New Service...** from the context menu and the **Service Properties** dialog box will be displayed
3. Specify a suitable name for the Service, for instance *http\_anti\_virus*
4. Make sure **TCP** is selected in the **Type** dropdown list
5. Select the HTTP ALG just created in the **ALG** dropdown list
6. Click the **TCP/UDP Parameters** tab
7. Enter **80** in the **Destination Port** textbox
8. Click **OK** in both of the open dialog boxes

C. Finally, modify the NAT rule to use the new service:

1. Go to **Rules**
2. Open the properties dialog box for the NAT rule handling the HTTP traffic between **lannet** and **all-nets**
3. Click the **Service** tab
4. Select your new service in the **Pre-defined** dropdown list
5. Click **OK**

Anti-Virus scanning is now activated for all web traffic from **lannet** to **all-nets**.

## 6.5. Web Content Filtering

### 6.5.1. Overview

Web traffic is one of the biggest sources for security issues and misuse of the Internet. Inappropriate surfing habits can expose a network to many security threats as well as legal and regulatory liabilities. Productivity and internet bandwidth can also be impaired.

CorePlus provides three mechanisms for filtering out web content that is deemed inappropriate for an organization or group of users:

- *Active Content Handling* can be used to "scrub" web pages of content that the administrator considers a potential threat, such as ActiveX objects and Java Applets.
- *Static Content Filtering* provides a means for manually classifying web sites as "good" or "bad". This is also known as URL *blacklisting* and *whitelisting*.
- *Dynamic Content Filtering* is a powerful feature that enables the administrator to allow or block access to web sites depending on the category they have been classified into by an automatic classification service. Dynamic content filtering requires a minimum of administration effort and has very high accuracy.

### 6.5.2. Active Content Handling

Some web content can contain malicious code designed to harm the workstation or the network from where the user is surfing. Typically, such code is embedded into various types of objects or files which are embedded into web pages.

CorePlus includes support for removing the following types of objects from web page content:

- ActiveX objects (including Flash)
- Java applets
- Javascript/VBScript code
- Cookies
- Invalidly formatted UTF-8 Characters (invalid URL formatting can be used to attack webservers)

The object types to be removed can be selected individually by configuring the corresponding HTTP Application Layer Gateway accordingly.



#### **Caution**

*Care should be taken before enabling removal of objects from web content.*

*Many web sites use Javascript and other types of client-side code and in most cases, the code is non-malicious. Common examples of this is the scripting used to implement drop-down menus as well as hiding and showing elements on web pages.*

*Removing such legitimate code could, at best, cause the web site to look distorted, at worst, cause it to not work in a browser at all. Active Content Handling should therefore only be used when the consequences are well understood.*

**Example 6.15. Stripping ActiveX and Java applets**

This example shows how to configure a HTTP Application Layer Gateway to strip ActiveX and Java applets. The example will use the `content_filtering` ALG object and presumes you have done one of the previous examples.

**Clavister FineTune**

1. Go to **Local Objects > Application Layer Gateways**
2. Right-click on the **content-filtering** object and choose **Properties**. The **Application Layer Gateway Properties** dialog box will be displayed.
3. Click the **Parameters...** button. The **HTTP Application Layer Gateway** dialog box will be shown.
4. Check the **Strip ActiveX objects (including flash)** control
5. Check the **Strip Java applets** control.
6. Click **OK** in both of the open dialog boxes.

## 6.5.3. Static Content Filtering

CorePlus can block or permit certain web pages based on configured lists of URLs which are called *blacklists* and *whitelists*. This type of filtering is also known as *Static Content Filtering*. The main benefit with Static Content Filtering is that it is an excellent tool to target specific web sites, and make the decision as to whether they should be blocked or allowed.

### Static and Dynamic Filter Ordering

Additionally, Static Content Filtering takes place *before* Dynamic Content Filtering (described below), which allows the possibility of manually making exceptions from the automatic dynamic classification process. In a scenario where goods have to be purchased from a particular on-line store, Dynamic Content Filtering might be set to prevent access to shopping sites by blocking the "Shopping" category. By entering the on-line store's URL into the HTTP Application Layer Gateway's whitelist, access to that URL is always allowed, taking precedence over Dynamic Content Filtering.

### Wildcarding

Both the URL blacklist and URL whitelist support wildcard matching of URLs in order to be more flexible. This wildcard matching is also applicable to the path following the URL hostname which means that filtering can be controlled to a file and directory level.

Below are some good and bad blacklist example URLs used for blocking:

<b>*.example.com/*</b>	<b>Good.</b> This will block all hosts in the <i>example.com</i> domain and all web pages served by those hosts.
<b>www.example.com/*</b>	<b>Good.</b> This will block the <i>www.example.com</i> website and all web pages served by that site.
<b>/*.*.gif</b>	<b>Good.</b> This will block all files with <i>.gif</i> as the file name extension.
<b>www.example.com</b>	<b>Bad.</b> This will only block the first request to the web site. Surfing to <i>www.example.com/index.html</i> , for instance, will not be blocked.
<b>*example.com/*</b>	<b>Bad.</b> This will also cause <i>www.myexample.com</i> to be blocked since it blocks all sites ending with <i>example.com</i> .

**Note**

Web content filtering URL blacklisting is a separate concept from Section 6.7, “Blacklisting Hosts and Networks”.

**Example 6.16. Setting up a white and blacklist**

This example shows the use of static content filtering where CorePlus can block or permit certain web pages based on blacklists and whitelists. As the usability of static content filtering will be illustrated, dynamic content filtering and active content handling will not be enabled in this example.

In this small scenario a general surfing policy prevents users from downloading .exe-files. However, the Clavister website provides secure and necessary program files which should be allowed to download.

**Clavister FineTune**

1. Go to **Local Objects > Application Layer Gateways**
2. Choose **New Application Layer Gateway...** from the context menu. The **Application Layer Gateway Properties** dialog box will be displayed.
3. Specify a suitable name for the ALG, for instance content\_filtering.
4. Select **HTTP** in the **Type** dropdown list.
5. Click the **Parameters...** button. The **HTTP Application Layer Gateway** dialog box will be shown.
6. Click the **URL Whitelist** tab.
7. In the whitelist control, enter `www.Clavister.com/*.exe`
8. Click the **URL Blacklist** tab.
9. In the blacklist control, enter `/*.exe`
10. Click **OK** in both of the open dialog boxes.

Simply continue adding specific blacklists and whitelists until the filter satisfies the needs.

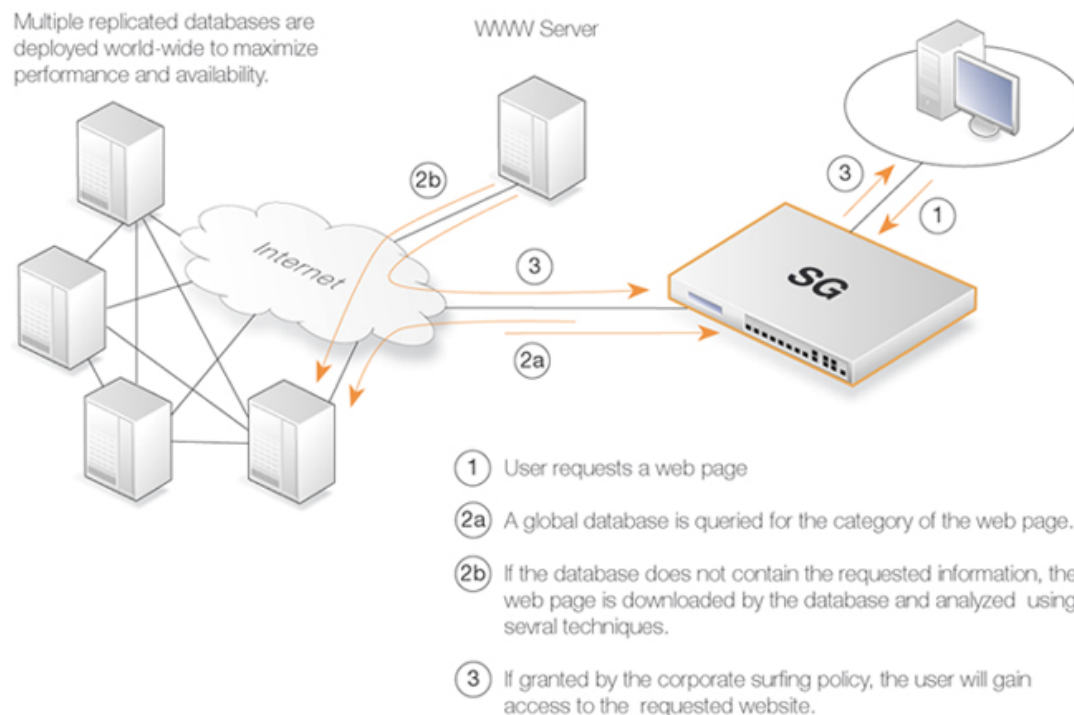
## 6.5.4. Dynamic Content Filtering

### 6.5.4.1. Overview

CorePlus supports *Dynamic Content Filtering* of web traffic, which enables an administrator to permit or block access to web pages based on the content of those web pages. This functionality is automated and it's not necessary to manually specify which URLs to block or allow. Instead, Clavister maintains a global infrastructure of databases containing massive numbers of current web site URL addresses, grouped into a variety of categories such as shopping, news, sport and adult-oriented on so on. These databases are updated every hour with new, categorized URLs while at the same time older, invalid URLs are dropped. The database content is global, covering websites in many different languages and which are hosted in countries around the world.

#### URL Processing Flow

When a user requests access to a web site, CorePlus sends a query to these databases to retrieve the category of the requested site. The user is then granted or denied access to the site based on the filtering policy in place for that category. If access is denied, a web page will be presented to the user explaining that the requested site has been blocked. To make the lookup process as fast as possible CorePlus maintains a local cache of recently accessed URLs. Caching can be highly efficient since a given user community, such as a group of university students, often surfs to a limited range of websites.

**Figure 6.2. Dynamic Content Filtering Flow**

If the requested web page URL is not present in the databases, then the webpage content at the URL will automatically be downloaded to Clavister's central data warehouse and automatically analyzed using a combination of techniques including neural networks and pattern matching. Once categorized, the URL is distributed to the global databases and CorePlus receives the category for the URL. Dynamic Content Filtering therefore requires a minimum of administration effort.

**Note**

*New, uncategorized URLs sent to the Clavister network are treated as anonymous submissions and no record of the source of new submissions is kept.*

**Categorizing Pages and Not Sites**

CorePlus dynamic filtering categorizes web pages and not sites. In other words, a web site may contain particular pages that should be blocked without blocking the entire site. CorePlus provides blocking down to the page level so that users may still access parts of websites that aren't blocked by the filtering policy.

**Enabling Dynamic Content Filtering**

Dynamic Content Filtering is a feature that is enabled by taking out a separate subscription to the service. This is an addition to the normal CorePlus license. For complete details of subscription services please see Appendix A, *Subscribing to Security Updates*.

Once a subscription is taken out, content filtering is enabled through the HTTP Application Layer Gateway (ALG) in combination with Services and the IP rule-set. This makes possible the setting up of a detailed content filtering policy based on the filtering parameters that are used for rules in the IP rule-set.

**Tip**

*If you would like your content filtering policy to vary depending on the time of the day,*



*make use of a schedule object in the corresponding IP rule. For more information, please see Section 3.6, “Schedules”.*

### Example 6.17. Enable Dynamic Content Filtering

This example shows how to setup a dynamic content filtering policy for HTTP traffic from **intnet** to **all-nets**. The policy will be configured to block all search sites, and this example assumes that the system is using a single NAT rule for HTTP traffic from **intnet** to **all-nets**.

#### **Clavister FineTune**

First, create an HTTP Application Layer Gateway (ALG) Object:

1. Go to **Local Objects > Application Layer Gateways**
2. Choose **New Application Layer Gateway...** from the context menu. The **Application Layer Gateway Properties** dialog box will be displayed.
3. Specify a suitable name for the ALG, for instance `content_filtering`
4. Select **HTTP** in the **Type** dropdown list.
5. Click the **Parameters...** button. The **HTTP Application Layer Gateway** dialog box will be shown.
6. Click the **Dynamic Content Filtering** tab.
7. Select **Enabled** in the **Mode** dropdown list.
8. In the **Blocked Categories** list, select **Search Sites** and click the **>>** button.
9. Click **OK** in both of the open dialog boxes.

Then, create a Service object using the new HTTP ALG:

1. Go to **Local Objects > Services**
2. Choose **New Service...** from the context menu. The **Service Properties** dialog box will be displayed.
3. Specify a suitable name for the Service, for instance `http_content_filtering`
4. Make sure **TCP** is selected in the **Type** dropdown list.
5. Select the HTTP ALG you just created in the **ALG** dropdown list.
6. Click the **TCP/UDP Parameters** tab.
7. Enter **80** in the **Destination Port** textbox.
8. Click **OK** in both of the open dialog boxes.

Finally, modify the NAT rule to use the new service:

1. Go to **Rules**
2. Open the properties dialog box for the NAT rule handling your HTTP traffic.
3. Click the **Service** tab.
4. Select your new service in the **Pre-defined** dropdown list.
5. Click **OK**.

Dynamic content filtering is now activated for all web traffic from `lannet` to `all-nets`. Validate the functionality by following these steps:

1. On a workstation on the `lannet` network, launch a standard web browser.
2. Try to browse to a search site, for instance `www.google.com`.
3. If everything is configured correctly, your web browser will present a web page that informs you about that the requested site is blocked.

## 6.5.4.2. Audit Mode

In *Audit Mode*, the system will classify and log all surfing according to the content filtering policy, but restricted web sites will still be accessible to the users. This means the content filtering feature of CorePlus can then be used as an analysis tool to analysis what categories of websites are being accessed by a user community and how often.

After running in Audit Mode for some weeks, it's then easy to have a good understanding of surfing behaviour and also the potential time savings that can be made by enabling content filtering. It's recommended that the administrator gradually introduces the blocking of particular categories one at a time. This allows individual users time to get used to the notion that blocking exists and can avoid the widespread protests that might occur if everything is blocked at once. Gradual introduction also makes evaluation easier as to whether the goals of blocking are being met.

### Example 6.18. Enabling Audit Mode

This example is based on the same scenario as the previous example, but now with audit mode enabled.

#### **Clavister FineTune**

First, create an HTTP Application Layer Gateway (ALG) Object:

1. Go to **Local Objects > Application Layer Gateways**
2. Choose **New Application Layer Gateway...** from the context menu. The **Application Layer Gateway Properties** dialog box will be displayed.
3. Specify a suitable name for the ALG, for instance `content_filtering`
4. Select **HTTP** in the **Type** dropdown list.
5. Click the **Parameters...** button. The **HTTP Application Layer Gateway** dialog box will be shown.
6. Click the **Dynamic Content Filtering** tab.
7. Select **Audit** in the **Mode** dropdown list.
8. In the **Blocked Categories** list, select **Search Sites** and click the **>>** button.
9. Click **OK** in both of the open dialog boxes.

The steps to then create a Service object using the new HTTP ALG and modifying the NAT rule to use the new service, are described in the previous example.

## 6.5.4.3. Allowing Override

On some occasions, Active Content Filtering may prevent users carrying out legitimate tasks. Consider a stock broker dealing with on-line gaming companies. In his daily work, he might need to browse gambling web sites to conduct company assessments. If the corporate policy blocks gambling web-sites, he won't be able to do his job.

For this reason, CorePlus supports a feature called *Allow Override*. With this feature enabled, the content filtering component will present a warning to the user that he is about to enter a web site that is restricted according to the corporate policy, and that his visit to the web site will be logged. This page is known as the *restricted site notice*. The user is then free to continue to the URL, or abort the request to prevent being logged.

By enabling this functionality, only users that have a valid reason to visit inappropriate sites will normally do so. Other will avoid those sites due to the obvious risk of exposing their surfing habits.

**Caution**

Enabling override can result in a user being able to surf to sites that are linked to by the visited site.

### 6.5.4.4. Reclassification of Blocked Sites

As the process of classifying unknown web sites is automated, there is always a small risk that some sites are given an incorrect classification. CorePlus provides a mechanism for allowing users to manually propose a new classification of sites.

This mechanism can be enabled on a per-HTTP ALG level, which means that you can choose to enable this functionality for regular users or for a selected user group only.

If reclassification is enabled and a user requests a web site which is disallowed, the block web page will include a dropdown list containing all available categories. If the user believes the requested web site is wrongly classified, he can select a more appropriate category from the dropdown list and submit that as a proposal.

The URL to the requested web site as well as the proposed category will then be sent to Clavister's central data warehouse for manual inspection. That inspection may result in the web site being reclassified, either according to the category proposed or to a category which is felt to be correct.

#### Example 6.19. Reclassifying a blocked site

This example shows how a user may propose a reclassification of a web site if he believes it is wrongly classified. This mechanism is enabled on a per-HTTP ALG level basis.

**Clavister FineTune**

First, create an HTTP Application Layer Gateway (ALG) Object:

1. Go to **Local Objects > Application Layer Gateways**
2. Choose **New Application Layer Gateway...** from the context menu. The **Application Layer Gateway Properties** dialog box will be displayed.
3. Specify a suitable name for the ALG, for instance `content_filtering`
4. Select **HTTP** in the **Type** dropdown list.
5. Click the **Parameters...** button. The **HTTP Application Layer Gateway** dialog box will be shown.
6. Click the **Dynamic Content Filtering** tab.
7. Select **Audit** in the **Mode** dropdown list.
8. In the **Blocked Categories** list, select **Search Sites** and click the **>>** button.
9. Check the **Allow Reclassification** control.
10. Click **OK** in both of the open dialog boxes.

Then, continue setting up the service object and modifying the NAT rule as we have done in the previous examples.

Dynamic content filtering is now activated for all web traffic from lannet to all-nets and the user is able to propose reclassification of blocked sites. Validate the functionality by following these steps:

1. On a workstation on the lannet network, launch a standard web browser.
2. Try to browse to a search site, for instance `www.google.com`.
3. If everything is configured correctly, your web browser will present a block page where a dropdown list containing all available categories is included.
4. The user is now able to select a more proper category and propose a reclassification.

### 6.5.4.5. Customizing the Block Web Page

The web page presented to the user can be customized to your needs by uploading a package of HTML pages to the system.

### 6.5.4.6. Event Messages from Dynamic Content Filtering

Dynamic Content Filtering utilizes the `url_request` event message to log its activities. The parameters of this event message will contain information on whether the request was allowed or blocked, and what categories the web site was classified as. Also, the message includes parameters specifying if audit mode or a restricted site notice were in effect.

**Note**

*The `url_request` event message will only be generated if event message generation has been enabled in the "parent" IP Rule.*

### 6.5.4.7. Content Filtering Categories

This section lists all the categories used with Dynamic Content Filtering and describes the purpose of each category.

**Category 1: Adult Content**

A web site may be classified under the Adult Content category if its content includes the description or depiction of erotic or sexual acts or sexually oriented material such as pornography. Exceptions to this are web sites that contain information relating to sexuality and sexual health, which may be classified under the Health Sites Category (21). Examples might be:

- [www.naughtychix.com](http://www.naughtychix.com)
- [www.fullonxxx.com](http://www.fullonxxx.com)

**Category 2: News**

A web site may be classified under the News category if its content includes information articles on recent events pertaining to topics surrounding a locality (eg. a town, city or nation) or culture, including weather forecasting information. Typically this would include most real-time online news publications and technology or trade journals. This does not include financial quotes, refer to the Investment Sites category (11), or sports, refer to the Sports category (16). Examples might be:

- [www.newsunlimited.com](http://www.newsunlimited.com)
- [www.dailyscoop.com](http://www.dailyscoop.com)

**Category 3: Job Search**

A web site may be classified under the Job Search category if its content includes facilities to search for or submit online employment applications. This also includes resume writing and posting and interviews, as well as staff recruitment and training services. Examples might be:

- [www.allthejobs.com](http://www.allthejobs.com)

- [www.yourcareer.com](http://www.yourcareer.com)

#### **Category 4: Gambling**

A web site may be classified under the Gambling category if its content includes advertisement or encouragement of, or facilities allowing for the partaking of any form of gambling; For money or otherwise. This includes online gaming, bookmaker odds and lottery web sites. This does not include traditional or computer based games; refer to the Games Sites category (10). Examples might be:

- [www.blackjackspot.com](http://www.blackjackspot.com)
- [www.pickapony.net](http://www.pickapony.net)

#### **Category 5: Travel / Tourism**

A web site may be classified under the Travel / Tourism category if its content includes information relating to travel activities including travelling for recreation and travel reservation facilities. Examples might be:

- [www.flythere.nu](http://www.flythere.nu)
- [www.reallycheaptix.com.au](http://www.reallycheaptix.com.au)

#### **Category 6: Shopping**

A web site may be classified under the Shopping category if its content includes any form of advertisement of goods or services to be exchanged for money, and may also include the facilities to perform that transaction online. Included in this category are market promotions, catalogue selling and merchandising services. Examples might be:

- [www.megamall.com](http://www.megamall.com)
- [www.buy-alcohol.se](http://www.buy-alcohol.se)

#### **Category 7: Entertainment**

A web site may be classified under the Entertainment category if its content includes any general form of entertainment that is not specifically covered by another category. Some examples of this are music sites, movies, hobbies, special interest, and fan clubs. This category also includes personal web pages such as those provided by ISPs. The following categories more specifically cover various entertainment content types, Pornography / Sex (1), Gambling (4), Chatrooms (8), Game Sites (10), Sports (16), Clubs and Societies (22) and Music Downloads (23). Examples might be:

- [www.celebnews.com](http://www.celebnews.com)
- [www.hollywoodlatest.com](http://www.hollywoodlatest.com)

#### **Category 8: Chatrooms**

A web site may be classified under the Chatrooms category if its content focuses on or includes real-time on-line interactive discussion groups. This also includes bulletin boards, message boards, on-line forums, discussion groups as well as URLs for downloading chat software. Examples might be:

- [www.thetalkroom.org](http://www.thetalkroom.org)
- [chat.yazoo.com](http://chat.yazoo.com)

### **Category 9: Dating Sites**

A web site may be classified under the Dating Sites category if its content includes facilities to submit and review personal advertisements, arrange romantic meetings with other people, mail order bride / foreign spouse introductions and escort services. Examples might be:

- [adultmatefinder.com](http://adultmatefinder.com)
- [www.marriagenow.com](http://www.marriagenow.com)

### **Category 10: Game Sites**

A web site may be classified under the Game Sites category if its content focuses on or includes the review of games, traditional or computer based, or incorporates the facilities for downloading computer game related software, or playing or participating in online games. Examples might be:

- [www.gamesunlimited.com](http://www.gamesunlimited.com)
- [www.gameplace.com](http://www.gameplace.com)

### **Category 11: Investment Sites**

A web site may be classified under the Investment Sites category if its content includes information, services or facilities pertaining to personal investment. URLs in this category include contents such as brokerage services, online portfolio setup, money management forums or stock quotes. This category does not include electronic banking facilities; refer to the E-Banking category (12). Examples might be:

- [www.loadsofmoney.com.au](http://www.loadsofmoney.com.au)
- [www.putsandcalls.com](http://www.putsandcalls.com)

### **Category 12: E-Banking**

A web site may be classified under the E-Banking category if its content includes electronic banking information or services. This category does not include Investment related content; refer to the Investment Sites category (11). Examples might be:

- [www.nateast.co.uk](http://www.nateast.co.uk)
- [www.borganfanley.com](http://www.borganfanley.com)

**Category 13: Crime / Terrorism**

A web site may be classified under the Crime / Terrorism category if its content includes the description, promotion or instruction in, criminal or terrorist activities, cultures or opinions. Examples might be:

- [www.beatthecrook.com](http://www.beatthecrook.com)

**Category 14: Personal Beliefs / Cults**

A web site may be classified under the Personal Beliefs / Cults category if its content includes the description or depiction of, or instruction in, systems of religious beliefs and practice. Examples might be:

- [www.paganfed.demon.co.uk](http://www.paganfed.demon.co.uk)
- [www.cultdeadcrow.com](http://www.cultdeadcrow.com)

**Category 15: Politics**

A web site may be classified under the Politics category if its content includes information or opinions of a political nature, electoral information and including political discussion groups. Examples might be:

- [www.democrats.org.au](http://www.democrats.org.au)
- [www.political.com](http://www.political.com)

**Category 16: Sports**

A web site may be classified under the Sports category if its content includes information or instructions relating to recreational or professional sports, or reviews on sporting events and sports scores. Examples might be:

- [www.sportstoday.com](http://www.sportstoday.com)
- [www.soccerball.com](http://www.soccerball.com)

**Category 17: www-Email Sites**

A web site may be classified under the www-Email Sites category if its content includes online, web-based email facilities. Examples might be:

- [www.coldmail.com](http://www.coldmail.com)
- [mail.yazoo.com](http://mail.yazoo.com)

**Category 18: Violence / Undesirable**

A web site may be classified under the Violence / Undesirable category if its contents are extremely violent or horrific in nature. This includes the promotion, description or depiction of violent acts, as well as web sites that have undesirable content and may not be classified elsewhere. Examples might be:

- [www.itstinks.com](http://www.itstinks.com)
- [www.ratemywaste.com](http://www.ratemywaste.com)

**Category 19: Malicious**

A web site may be classified under the Malicious category if its content is capable of causing damage to a computer or computer environment, including malicious consumption of network bandwidth. This category also includes "Phishing" URLs which designed to capture secret user authentication details by pretending to be a legitimate organisation. Examples might be:

- [hastalavista.baby.nu](http://hastalavista.baby.nu)

**Category 20: Search Sites**

A web site may be classified under the Search Sites category if its main focus is providing online Internet search facilities. Refer to the section on unique categories at the start of this document. Examples might be:

- [www.zoogle.com](http://www.zoogle.com)
- [www.yazoo.com](http://www.yazoo.com)

**Category 21: Health Sites**

A web site may be classified under the Health Sites category if its content includes health related information or services, including sexuality and sexual health, as well as support groups, hospital and surgical information and medical journals. Examples might be:

- [www.thehealthzone.com](http://www.thehealthzone.com)
- [www.safedrugs.com](http://www.safedrugs.com)

**Category 22: Clubs and Societies**

A web site may be classified under the Clubs and Societies category if its content includes information or services of relating to a club or society. This includes team or conference web sites. Examples might be:

- [www.sierra.org](http://www.sierra.org)
- [www.walkingclub.org](http://www.walkingclub.org)



**Category 23: Music Downloads**

A web site may be classified under the Music Downloads category if it provides online music downloading, uploading and sharing facilities as well as high bandwidth audio streaming. Examples might be:

- [www.onlymp3s.com](http://www.onlymp3s.com)
- [www.mp3space.com](http://www.mp3space.com)

**Category 24: Business Oriented**

A web site may be classified under the Business Oriented category if its content is relevant to general day-to-day business or proper functioning of the Internet, eg. Web browser updates. Access to web sites in this category would in most cases not be considered unproductive or inappropriate.

**Category 25: Government Blocking List**

This category is populated by URLs specified by a government agency, and contains URLs that are deemed unsuitable for viewing by the general public by way of their very extreme nature. Examples might be:

- [www.verynastystuff.com](http://www.verynastystuff.com)
- [www.unpleasantvids.com](http://www.unpleasantvids.com)

**Category 26: Educational**

A web site classified under the Educational category may belong to other categories but has content that relates to educational services or has been deemed of educational value, or to be an educational resource, by educational organisations. This category is populated by request or submission from various educational organisations. Examples might be:

- [highschoolsays.org](http://highschoolsays.org)
- [www.learn-at-home.com](http://www.learn-at-home.com)

**Category 27: Advertising**

A web site may be classified under the Advertising category if its main focus includes providing advertising related information or services. Examples might be:

- [www.admessages.com](http://www.admessages.com)
- [www.tripleclick.com](http://www.tripleclick.com)

**Category 28: Drugs/Alcohol**

A web site may be classified under the Drugs/Alcohol category if its content includes drug and alco-

hol related information or services. Some URLs categorised under this category may also be categorised under the Health category. Examples might be:

- [www.the-cocktail-guide.com](http://www.the-cocktail-guide.com)
- [www.stiffdrinks.com](http://www.stiffdrinks.com)

### **Category 29: Computing/IT**

A web site may be classified under the Computing/IT category if its content includes computing related information or services. Examples might be:

- [www.purplehat.com](http://www.purplehat.com)
- [www.gnu.org](http://www.gnu.org)

### **Category 30: Swimsuit/Lingerie/Models**

A web site may be categorised under the Swimsuit/Lingerie/Models category if its content includes information pertaining to, or images of swimsuit, lingerie or general fashion models. Examples might be:

- [www.vickys-secret.com](http://www.vickys-secret.com)
- [sportspictured.cnn.com/features/2002/swimsuit](http://sportspictured.cnn.com/features/2002/swimsuit)

### **Category 31: Spam**

A web site may be classified under the Spam category if it is found to be contained in bulk or spam emails. Examples might be:

- [kaqsovdij.gjibhgk.info](mailto:kaqsovdij.gjibhgk.info)
- [www.pleaseupdateyourdetails.com](http://www.pleaseupdateyourdetails.com)

### **Category 32: Non-Managed**

Unclassified sites and sites that don't fit one of the other categories will be placed in this category. It is unusual to block this category since this could result in most harmless URLs being blocked.

## 6.6. Denial-Of-Service Attacks (DoS)

By embracing the Internet enterprises experience new business opportunities and growth. The enterprise network and the applications that run over it have become business critical, an essential part of the organization's strategy to increase revenues and profits. Not only can a company reach a larger number of customers via the Internet, it can serve them faster and more efficiently. At the same time, using a public IP network enables companies to reduce infrastructure-related costs.

Unfortunately, the same advantages that the Internet brings to business also benefit the hackers who propagate their attack techniques via the same public infrastructure. Hacker attack tools are readily available on the Internet, including the development source code. Work on these tools is often split across groups of novice hackers — known as "script kiddies" or "larval hackers" — who are scattered across the globe, providing around-the-clock progression of automated attack methods. In addition, many of the new hacking methods utilize the distributed nature of the Internet to launch DoS attacks against unprotected organizations.

To be on the receiving end of a Distributed Denial of Service (DoS) attack is probably the last thing any security specialist or network administrator wants to experience. Most of us have probably read about the high-profiled DoS assaults on large and successful companies and their relentless struggles with machines going down, jammed Internet connections and business critical systems being overloaded. These attacks seem to appear out of nothing and the consequences has been devastating. Not only does it ruin the business financially as a direct consequence but it also vandalize the reputation and trust account for the company thus resulting in financial loss indirectly in the long-term perspective as well. The ever-growing number and types of DoS and Distributed Denial of Service (DDoS) attacks continue to bring down worldwide networks at an alarming rate. Within the last year, nearly 40 percent of all businesses with an Internet presence experienced at least one DoS attack, with a total cost in terms of lost business and shaken customer confidence in hundreds of millions of dollars.

In this section you can learn how to use the Clavister Security Gateway to protect your organization against these sophisticated Denial of Service attacks.

### 6.6.1. About Denial of Service attacks

A DoS attack can be perpetrated in a number of ways but there are three basic types of attack:

- consumption of computational resources, such as bandwidth, disk space, or CPU time
- disruption of configuration information, such as routing information
- disruption of physical network components

One of the most commonly used method is the consumption of computational resources which means that the DoS attack floods the network and ties up critical resources used to run business critical applications. In some cases, vulnerabilities in the Unix and Windows operating systems are exploited to intentionally crash the system, while in other cases large amounts of apparently valid traffic are directed at sites until they become overloaded and crash.

The most commonly used DoS attacks are:

- The Ping of Death / Jolt attacks
- Fragmentation overlap attacks: Teardrop / Bonk / Boink / Nostea
- The Land and LaTierra attacks
- The WinNuke attack
- Amplification attacks: Smurf, Papasmurf, Fraggle

- The TCP SYN Flood attack
- The Jolt2 attack

## 6.6.2. The Ping of Death and Jolt attacks

The "ping of death" is one of the earliest layer 3/4 attacks. One of the simplest ways to execute it is to run "ping -l 65510 1.2.3.4" on a Windows 95 system where 1.2.3.4 is the IP address of the intended victim. "Jolt" is simply a purpose-written program for generating such packets on operating systems whose ping commands refuse to generate oversized packets.

The triggering factor is that the last fragment makes the total packet size exceed 65535 bytes, which is the highest number that a 16-bit integer can store. When the value overflows, it jumps back to a very small number. What happens then is a function of how well the victim's IP stack is implemented.

CorePlus will never allow fragments through that would result in the total size exceeding 65535 bytes. In addition to that, there are configurable limits for IP packet sizes in the "Advanced Settings" section.

Ping of death will show up in CorePlus logs as drops with the rule name set to "LogOversizedPackets". The sender IP address may be spoofed.

## 6.6.3. Fragmentation overlap attacks: Teardrop, Bonk, Boink and Nestea

Teardrop and its followers are fragment overlap attack. Many IP stacks have shown erratic behavior (excessive resource exhaustion or crashes) when exposed to overlapping fragments.

CorePlus protects fully against fragmentation overlap attacks. Overlapping fragments are never allowed to pass through the system.

Teardrop and its followers will show up in CorePlus logs as drops with the rule name set to "Illegal-Frags". The sender IP address may be spoofed.

## 6.6.4. The Land and LaTierra attacks

The Land and LaTierra attacks works by sending a packet to a victim and making the victim respond back to itself, which in turn generates yet another response to itself, etc etc. This will either bog the victim's machine down, or make it crash.

The attack is accomplished by using the victim's IP address in the source field of an IP packet as well as in the destination field.

CorePlus protects against this attack by applying IP spoofing protection to all packets. In its default configuration, it will simply compare arriving packets to the contents of the routing table; if a packet arrives on an interface that is different from the interface where the system expects the source to be, the packet will be dropped.

Land and LaTierra attacks will show up in CorePlus logs as drops with the rule name set to "AutoAccess" by default, or, if you have written custom Access rules, the name of the Access rule that dropped the packet. The sender IP address is of no interest; it is always the same as the destination IP address.

## 6.6.5. The WinNuke attack

The WinNuke attack works by connecting to a TCP service that does not have handlers for "out-of-band" data (TCP segments with the URG bit set), but still accepts such data. This will usually put the service in a tight loop that consumes all available CPU time.

One such service was the NetBIOS over TCP/IP service on Windows machines, which gave the attack its name.

CorePlus protects against this in two ways:

- With a careful inbound policy, the attack surface is greatly reduced. Only exposed services could possibly become victims to the attack, and public services tend to be more well-written than services expected to only serve the local network.
- By stripping the URG bit by default from all TCP segments traversing the system (configurable via **Advanced Settings > TCP > TCPUrg**).

WinNuke attacks will usually show up in CorePlus logs as normal drops with the name of the rule in your policy that disallowed the connection attempt. For connections allowed through the system, "TCP" or "DROP" category (depending on the TCPUrg setting) entries will appear, with a rule name of "TCPUrg". The sender IP address is not likely to be spoofed; a full three-way handshake must be completed before out-of-band segments can be sent.

## 6.6.6. Amplification attacks: Smurf, Papasmurf, Fraggle

This category of attacks all make use of "amplifiers": poorly configured networks who amplify a stream of packets and send it to the ultimate target. The goal is excessive bandwidth consumption - consuming all of the victim's Internet connection capacity. An attacker with sufficient bandwidth can forgo the entire amplification stage and simply stream enough bandwidth at the victim. However, these attacks allows attackers with less bandwidth than the victim to amplify their data stream to overwhelm the victim.

- "Smurf" and "Papasmurf" send ICMP echo packets to the broadcast address of open networks with many machines, faking the source IP address to be that of the victim. All machines on the open network then "respond" to the victim.
- "Fraggle" uses the same general idea, but instead using UDP echo (port 7) to accomplish the task. Fraggle generally gets lower amplification factors since there are fewer hosts on the Internet that have the UDP echo service enabled.

Smurf attacks will show up in CorePlus logs as masses of dropped ICMP Echo Reply packets. The source IP addresses will be those of the amplifier networks used. Fraggle attacks will show up in CorePlus logs as masses of dropped (or allowed, depending on policy) packets. The source IP addresses will be those of the amplifier networks used.

### Avoiding becoming an amplifier

Even though the brunt of the bandwidth stream is at the ultimate victim's side, being selected as an amplifier network can also consume great resources. In its default configuration, CorePlus explicitly drops packets sent to broadcast address of directly connected networks (configurable via **Advanced Settings > IP > DirectedBroadcasts**). However, with a reasonable inbound policy, no protected network should ever have to worry about becoming a smurf amplifier.

### Protection at the ultimate victim side

Smurf, and its followers, are resource exhaustion attacks. More specifically: they exhaust your Internet connection. In the general case, the gateway is situated at the "wrong" side of the Internet connection bottleneck to provide much protection against this class of attacks. The damage has already been done by the time the packets reach the gateway.

However, CorePlus may be of some help in keeping the load off of internal servers, making them

available for internal service, or perhaps service via a secondary Internet connection not targeted by the attack.

- Smurf and Papasmurf floods will be seen as ICMP Echo Responses at the victim side. Unless "FwdFast" rules are in use, such packets are never allowed to initiate new connections, regardless of whether or not there are rules that allow the traffic.
- Fraggle packets may arrive at any UDP destination port of the attacker's discretion. Tightening ones inbound ruleset may help.

The Traffic Shaping feature built into CorePlus also help absorb some of the flood before it reaches protected servers.

## 6.6.7. The TCP SYN Flood attack

The TCP SYN Flood attack works by sending large amounts of TCP SYN packets to a given port and then not responding to SYN ACKs sent in response. This will tie up local TCP stack resources on the victim machine until it is unable to respond to more SYN packets until the existing half-open connections have timed out.

CorePlus will protect against TCP SYN Flood attacks if "SynRelay" is enabled in the rule or service allowing the traffic. By default, this is the case for the "http-in", "https-in", "smtp-in", and "ssh-in" services.

The "SynRelay" protection works by completing the 3-way handshake with the client before doing a second handshake of its own with the target service. Overload situations do not occur nearly as easily in CorePlus due to much better resource management and lack of restrictions normally placed upon a full-blown operating system. While a normal operating system can exhibit problems with as few as 5 outstanding half-open connections, CorePlus can fill its entire state table (thousands or millions of connections, depending on your product model), before anything out of the ordinary happens. When the state table fills up, old outstanding SYN connections will be among the first to be dropped to make room for new connections.

TCP SYN Flood attacks will show up in CorePlus logs as excessive amounts of new connections (or drops, if the attack is targeted at a closed port). The sender IP address is almost invariably spoofed.

## 6.6.8. The Jolt2 attack

The Jolt2 attack works by sending a steady stream of identical fragments at the victim machine. A few hundred packets per second will freeze vulnerable machines completely until the stream is ended.

CorePlus will protect completely against this attack. The first fragment will be enqueued, waiting for earlier fragments to arrive so that they may be passed on in order, but this never happens, so not even the first fragment gets through. Subsequent fragments will be thrown away as they are identical to the first fragment.

If the attacker chooses a fragment offset higher than the limits imposed by the **Advanced Settings > LengthLim** in CorePlus, the packets will not even get that far; they will be dropped immediately. Jolt2 attacks may or may not show up in CorePlus logs. If the attacker chooses a too-high fragment offset for the attack, they will show up as drops from the rule-set to "LogOversizedPackets". If the fragment offset is low enough, no logging will occur. The sender IP address may be spoofed.

## 6.6.9. Distributed Denial of Service attacks

A more sophisticated form of DoS is the Distributed Denial of Service (DDoS) attack. DDoS attacks involve breaking into hundreds or thousands of machines all over the Internet to install DDoS software on them, allowing the hacker to control all these burgled machines to launch coordinated attacks on victim sites. These attacks typically exhaust bandwidth, router processing capacity, or network stack resources, breaking network connectivity to the victims.

Although recent DDoS attacks have been launched from both private corporate and public institutional systems, hackers tend to favor university networks because of their open, distributed nature. Tools used to launch DDoS attacks include Trin00, TribeFlood Network (TFN), TFN2K and Stacheldraht.

## 6.7. Blacklisting Hosts and Networks

CorePlus implements a *Blacklist* of host or network IP addresses which can be utilized to protect against traffic coming from specific internet sources.

Certain CorePlus modules, specifically the Intrusion Detection and Prevention (IDP) module, as well as the Rate Limiting feature of Threshold Limits, can make use of this Blacklist when certain conditions are encountered, such as traffic triggering a rate limiting rule.

Adding a host or network to the Blacklist can be enabled in IDP and in Rate Limiting by specifying the **Protect** action for when a rule is triggered. Once enabled there are 3 Blacklisting options:

<b>Time to Block Host/Network in seconds</b>	The host or network which is the source of the traffic will stay on the blacklist for the specified time and then be removed. If the same source triggers another entry to the blacklist then the blocking time is renewed to it's original, full value (ie. it is not cumulative).
<b>Block only this Service</b>	By default Blacklisting blocks all Services for the triggering host.
<b>Exempt already established connections from Blacklisting</b>	If there are established connections that have the same source as this new Blacklist entry then they won't be dropped if this option is set.

IP addresses or networks are added to the list and the traffic from these sources is then blocked for a period of time. The Blacklist is maintained even if the Clavister Security Gateway shuts down or re-boots.

### Whitelisting

To ensure that "good" internet traffic sources are not blocked under any circumstances, a *Whitelist* is also maintained by CorePlus. It can be advisable to add the Clavister Security Gateway itself to the Whitelist as well as the IP addresses of all management workstations.

For further details on usage see Section 6.3.5, "IDP Actions" and Section 10.2.1, "Rate limit blacklisting".



#### **Note**

*Content filtering blacklisting is a separate subject and uses a separate logical list (see Section 6.5, "Web Content Filtering").*

#### **Example 6.20. Adding a whitelist entry**

##### **Clavister FineTune**

1. Go to **Miscellaneous > Whitelist**
2. Select **New Whitelist item** from the context menu and the **Whitelist Item Properties** dialog will appear.
3. Now Enter:
  - a. **IP Address:** Choose the gateway's own LAN IP address from the dropdown list
  - b. **Protocol:** Select **All** from the dropdown list
4. Click **OK**





---

# Chapter 7. Address Translation

This chapter describes CorePlus address translation capabilities.

- Dynamic Address Translation (NAT), page 165
- Static Address Translation (SAT), page 168

CorePlus supports two types of address translation: Dynamic Address Translation (NAT) and Static Address Translation (SAT). Both types of translations are policy-based, and can thus be applied on any type of traffic through the system. Two specific types of rules, NAT and SAT rules, are used to specify address translation policies within the standard IP rule-set.

There are two main reasons for employing address translation:

- **Functionality.** Perhaps you use private IP addresses on your protected network and your protected hosts to have access to the Internet. This is where dynamic address translation may be used. You might also have servers with private IP addresses that need to be publicly accessible. This is where static address translation may be of assistance.
- **Security.** Address translation does not in and of itself provide any greater level of security, but it can make it more difficult for intruders to understand the exact layout of your protected network and which machines are susceptible to attack. In the worst case scenario, employing address translation will mean that an intruders attack will take longer, which will also make him more visible in the gateways log files. In the best-case scenario, the intruder will just give up.

This section describes dynamic as well as static address translation, how they work and what they can and cannot do. It also provides examples of what NAT and SAT rules can look like.

## 7.1. Dynamic Address Translation (NAT)

Dynamic Address Translation (hereinafter referred to as NAT) provides a method for translating the original source IP address to a different address. The most common usage for NAT is when you are using private IP addresses on one of your internal networks, and would like the outbound connections to appear as they are originating from the Clavister Security Gateway itself.

NAT is a many-to-one translation, meaning that each NAT rule will translate several source IP addresses into a single source IP address. To maintain session state information, each connection from dynamically translated addresses must use a unique port number and IP address combination as its sender. Therefore, CorePlus will perform an automatic translation of the source port number as well. The source port used will be the next free port, usually one above 32768. This means that there is a limitation of about 30000 simultaneous connections using the same translated source IP address.

CorePlus supports two strategies for how to translate the source address:

### **Use Interface Address**

When a new connection is established, the routing table is consulted to resolve the egress interface for that connection. The IP address of that resolved interface is then being used as the new source IP address when CorePlus performs the address translation.

### **Specify Sender Address**

A specific IP address can be specified as the new source IP address. The specified IP address needs to have a matching ARP Publish entry configured for the egress interface. Otherwise, the return traffic will not be received by the Clavister Security Gateway.

The following example illustrates how NAT is applied in practice on a new connection:

1. The sender, e.g. 192.168.1.5, sends a packet from a dynamically assigned port, for instance, port 1038, to a server, e.g. 195.55.66.77 port 80.

**192.168.1.5:1038 => 195.55.66.77:80**

2. In this example, the Use Interface Address option is used, and we will use 195.11.22.33 as the interface address. In addition, the source port is changed to a free port on the Clavister Security Gateway, usually one above 32768. In this example, we will use port 32789. The packet is then sent to its destination.

**195.11.22.33:32789 => 195.55.66.77:80**

3. The recipient server then processes the packet and sends its response.

**195.55.66.77:80 => 195.11.22.33:32789**

4. CorePlus receives the packet and compares it to its list of open connections. Once it finds the connection in question, it restores the original address and forwards the packet.

**195.55.66.77:80 => 192.168.1.5:1038**

5. The original sender receives the response.

#### Example 7.1. Adding a NAT Policy

To add a policy that will perform address translation for all HTTP traffic originating from the internal network, follow the steps outlined below:

##### *Clavister FineTune*

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance NAT\_HTTP.
4. Now enter:
  - **Action:** NAT
  - **Source Interface:** lan
  - **Source Network:** lannet
  - **Destination Interface:** any
  - **Destination Network:** all-nets
5. Under the **Service** tab, select **http** in the **Pre-defined** dropdown list.
6. Under the **Address Translation** tab, make sure that the **Use Interface Address** option is selected.
7. Click **OK**.

## 7.1.1. Which Protocols can NAT handle?

Dynamic address translation is able to handle the TCP, UDP and ICMP protocols with a good level of functionality since the algorithm knows of values that can be adjusted to become unique in the three protocols. For other IP level protocols, unique connections are identified by their sender addresses, destination addresses and protocol numbers.

This means that:

- An internal machine can communicate with several external servers using the same IP protocol.
- An internal machine can communicate with several external servers using different IP protocols.
- Several internal machines can communicate with different external server using the same IP protocol.
- Several internal machines can communicate with the same server using different IP protocols.
- Several internal machines can *not* communicate with the same external server using the same IP protocol.



**Note**

*These restrictions apply only to IP level protocols other than TCP, UDP and ICMP, e.g. OSPF, L2TP, etc. They do not apply to "protocols" transported by TCP, UDP and ICMP such as telnet, FTP, HTTP, SMTP, etc. CorePlus can alter port number information in the TCP and UDP headers to make each connection unique, even though such connections have had their sender addresses translated to the same IP.*

Some protocols, regardless of the method of transportation used, can cause problems during address translation.

## 7.2. Static Address Translation (SAT)

CorePlus can translate entire ranges of IP addresses and/or ports. Such translations are transpositions, that is, each address or port is mapped to a corresponding address or port in the new range, rather than translating them all to the same address or port. This functionality is known as Static Address Translation, hereinafter referred to as SAT.

Unlike NAT, a SAT policy requires more than a single SAT rule to function. CorePlus does not terminate the rule-set lookup upon finding a matching SAT rule. Instead, it continues to search for a matching Allow, NAT or FwdFast rule. Only when it has found such a matching rule does the system execute the static address translation.

### 7.2.1. Translation of a Single IP Address (1:1)

The simplest form of SAT is translation of a single IP address. A very common usage for this type of SAT is to enable external users to access a protected server having a private address. This scenario is also commonly referred to as *Virtual IP* or *Virtual Server* in other types of products.

#### Example 7.2. Enabling Traffic to a Protected Web Server in a DMZ

In this example, we will create a SAT policy that will translate and allow connections from the Internet to a web server located in a DMZ. The Clavister Security Gateway is connected to the Internet using the wan interface with address object ip\_wan (defined as 195.55.66.77) as IP address. The web server has the IP address 10.10.10.5 and is reachable through the dmz interface.

##### **Clavister FineTune**

First create a SAT rule:

1. Select the **Rules** section of the target system in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance SAT\_HTTP\_To\_DMZ.
4. Now enter:
  - **Action:** SAT
  - **Source Interface:** any
  - **Source Network:** all-nets
  - **Destination Interface:** core
  - **Destination Network:** ip\_wan
5. Under the **Service** tab, select **http** in the **Pre-defined** dropdown list.
6. Switch to the **Address Translation** tab.
7. Make sure that the **Destination IP Address** option is selected.
8. In the **New IP Address** textbox, enter 10.10.10.5
9. Click **OK**.

Then create a corresponding Allow rule:

1. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
2. Specify a suitable name for the rule, for instance Allow\_HTTP\_To\_DMZ.
3. Now enter:
  - **Action:** Allow
  - **Source Interface:** any

- **Source Network:** all-nets
  - **Destination Interface:** core
  - **Destination Network:** ip\_wan
4. Under the **Service** tab, select **http** in the **Pre-defined** dropdown list.
  5. Click **OK**.

The example results in the following two rules in the rule-set:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	ip_wan	http SETDEST 10.10.10.5 80
2	Allow	any	all-nets	core	ip_wan	http

These two rules allow us to access the web server via the Clavister Security Gateway's external IP address. Rule 1 states that address translation can take place if the connection has been permitted, and rule 2 permits the connection.

Of course, we also need a rule that allows internal machines to be dynamically address translated to the Internet. In this example, we use a rule that permits everything from the internal network to access the Internet via NAT hide:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
3	NAT	lan	lannet	any	all-nets	All

Now, what is wrong with this rule-set?

Well, if we assume that we want to implement address translation for reasons of security as well as functionality, we discover that this rule-set makes our internal addresses visible to machines in the DMZ. When internal machines connect to ip\_wan port 80, they will be allowed to proceed by rule 2 as it matches that communication. From an internal perspective, all machines in the DMZ should be regarded as any other Internet-connected servers; we do not trust them, which is the reason for locating them in a DMZ in the first place.

There are two possible solutions:

1. You can change rule 2 so that it only applies to external traffic.
2. You can swap rules 2 and 3 so that the NAT rule is carried out for internal traffic before the Allow rule matches.

Which of these two options is the best?

For this configuration, it makes no difference whatsoever. Both solutions work just as well.

However, suppose that we use another interface, ext2, in the Clavister Security Gateway and connect it to another network, perhaps to that of a neighboring company so that they can communicate much faster with our servers.

If option 1 was selected, the rule-set must be adjusted thus:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	ip_wan	http SETDEST 10.10.10.5 80
2	Allow	wan	all-nets	core	ip_wan	http
3	Allow	ext2	ext2net	core	ip_wan	http
4	NAT	lan	lannet	any	all-nets	All

This increases the number of rules for each interface allowed to communicate with the web server. However, the rule ordering is unimportant, which may help avoid errors.

If option 2 was selected, the rule-set must be adjusted thus:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	ip_wan	http SETDEST

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
						10.10.10.5 80
2	NAT	lan	lannet	any	all-nets	All
3	Allow	any	all-nets	core	ip_wan	http

This means that the number of rules does not need to be increased. This is good as long as all interfaces can be entrusted to communicate with the web server. However, if, at a later point, you add an interface that cannot be entrusted to communicate with the web server, separate Drop rules would have to be placed before the rule granting all machines access to the web server.

Determining the best course of action must be done on a case-by-case basis, taking all circumstances into account.

### Example 7.3. Enabling Traffic to a Web Server on an Internal Network

The example we have decided to use is that of a web server with a private address located on an internal network. From a security standpoint, this approach is wrong, as web servers are very vulnerable to attack and should therefore be located in a DMZ. However, due to its simplicity, we have chosen to use this model in our example.

In order for external users to access the web server, they must be able to contact it using a public address. In this example, we have chosen to translate port 80 on the Clavister Security Gateway's external address to port 80 on the web server:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	ip_wan	http SETDEST wwwsrv 80
2	Allow	any	all-nets	core	ip_wan	http

These two rules allow us to access the web server via the Clavister Security Gateway's external IP address. Rule 1 states that address translation can take place if the connection has been permitted, and rule 2 permits the connection.

Of course, we also need a rule that allows internal machines to be dynamically address translated to the Internet. In this example, we use a rule that permits everything from the internal network to access the Internet via NAT hide:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
3	NAT	lan	lannet	any	all-nets	All

The problem with this rule-set is that it will not work at all for traffic from the internal network.

In order to illustrate exactly what happens, we use the following IP addresses:

- ip\_wan (195.55.66.77): a public IP address
- ip\_lan (10.0.0.1): the Clavister Security Gateway's private internal IP address
- wwwsrv (10.0.0.2): the web servers private IP address
- PC1 (10.0.0.3): a machine with a private IP address
- PC1 sends a packet to ip\_wan to reach "www.ourcompany.com":  
**10.0.0.3:1038 => 195.55.66.77:80**
- CorePlus translates the address in accordance with rule 1 and forwards the packet in accordance with rule 2:  
**10.0.0.3:1038 => 10.0.0.2:80**
- wwwsrv processes the packet and replies:  
**10.0.0.2:80 => 10.0.0.3:1038**

This reply arrives directly to PC1 without passing through the Clavister Security Gateway. This causes problems. The reason this will not work is because PC1 expects a reply from 195.55.66.77:80, not 10.0.0.2:80. The unexpected reply is discarded and PC1 continues to wait for a response from 195.55.66.77:80, which will never arrive.

Making a minor change to the rule-set in the same way as described above, will solve the problem. In this ex-

ample, for no particular reason, we choose to use option 2:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	ip_wan	http SETDEST wwwsrv 80
2	NAT	lan	lanet	any	all-nets	All
3	Allow	any	all-nets	core	ip_wan	http

- PC1 sends a packet to ip\_wan to reach "www.ourcompany.com":  
**10.0.0.3:1038 => 195.55.66.77:80**
- CorePlus address translates this statically in accordance with rule 1 and dynamically in accordance with rule 2:  
**10.0.0.1:32789 => 10.0.0.2:80**
- wwwsrv processes the packet and replies:  
**10.0.0.2:80 => 10.0.0.1:32789**
- The reply arrives and both address translations are restored:  
**195.55.66.77:80 => 10.0.0.3:1038**

This way, the reply arrives at PC1 from the expected address.

Another possible solution to this problem is to allow internal clients to speak directly to 10.0.0.2, which would completely avoid all the problems associated with address translation. However, this is not always practical.

## 7.2.2. Translation of Multiple IP Addresses (M:N)

A single SAT rule can be used to translate an entire range of IP addresses. In this case, the result is a transposition of addresses, where the first original IP address will be translated to the first IP address in the translation list and so forth.

For instance, a SAT policy specifying that connections to the 194.1.2.16/29 network should be translated to 192.168.0.50 will result in transpositions as per the table below:

Original Address	Translated Address
194.1.2.16	192.168.0.50
194.1.2.17	192.168.0.51
194.1.2.18	192.168.0.52
194.1.2.19	192.168.0.53
194.1.2.20	192.168.0.54
194.1.2.21	192.168.0.55
194.1.2.22	192.168.0.56
194.1.2.23	192.168.0.57

In other words:

- Attempts to communicate with 194.1.2.16 will result in a connection to 192.168.0.50.
- Attempts to communicate with 194.1.2.22 will result in a connection to 192.168.0.56.

An example of when this is useful is when having several protected servers in a DMZ, and where each server should be accessible using a unique public IP address.

### Example 7.4. Translating Traffic to Multiple Protected Web Servers

In this example, we will create a SAT policy that will translate and allow connections from the Internet to five web servers located in a DMZ. The Clavister Security Gateway is connected to the Internet using the wan interface,



and the public IP addresses to use are in the range of 195.55.66.77 to 195.55.66.81. The web servers have IP addresses in the range 10.10.10.5 to 10.10.10.9, and they are reachable through the dmz interface.

To accomplish the task, the following steps need to be performed:

- Define an address object containing the public IP addresses.
- Define another address object for the base of the web server IP addresses.
- Publish the public IP addresses on the wan interface using the ARP publish mechanism.
- Create a SAT rule that will perform the translation.
- Create an Allow rule that will permit the incoming HTTP connections.

### **Clavister FineTune**

Create an address object for the public IP addresses:

1. Select the **Local Objects/Hosts & Networks** section in the tree view of the Security Editor.
2. Choose **New Host & Network...** from the context menu, the **Host & Network Properties** dialog box will be displayed.
3. Specify a suitable name for the object, for instance `wwsrvv_pub`.
4. Select **Range** as type.
5. Now enter:
  - **IP Low:** 195.55.66.77
  - **IP High:** 195.55.66.81
6. Click **OK**.

Now, create another address object for the base of the web server IP addresses:

1. Choose **New Host & Network...** from the context menu, the **Host & Network Properties** dialog box will be displayed.
2. Specify a suitable name for the object, for instance `wwsrvv_priv_base`.
3. Select **Host** as type.
4. Enter 10.10.10.5 in the **IP Address** edit box.
5. Click **OK**.

Publish the public IP addresses on the wan interface using ARP publish. One ARP item is needed for every IP address:

1. Select the **Interfaces/ARP** section in the tree view of the Security Editor.
2. Choose **New ARP...** from the context menu. The **ARP Properties** dialog box will be displayed.
3. Now enter:
  - **Mode:** Publish
  - **Interface:** wan
  - **IP Address:** 195.55.66.77
4. Click **OK** and repeat for all the five public IP addresses.

Create a SAT rule for the translation:

1. Select the **Rules** section in the tree view of the Security Editor.
2. Choose **New Rule...** from the context menu. The **Rule Properties** dialog box will be displayed.
3. Specify a suitable name for the rule, for instance `SAT_HTTP_To_DMZ`.

4. Now enter:
  - **Action:** SAT
  - **Source Interface:** any
  - **Source Network:** all-nets
  - **Destination Interface:** core
  - **Destination Network:** wwwsrv\_pub
5. Under the **Service** tab, select **http** in the **Pre-defined** dropdown list.
6. Switch to the **Address Translation** tab.
7. Make sure that the **Destination IP Address** option is selected.
8. In the **New IP Address** dropdown list, select **wwwsrv\_priv**.
9. Click **OK**.

Finally, create a corresponding Allow rule:

1. Choose **New Rule...** from the context menu, the **Rule Properties** dialog box will be displayed.
2. Specify a suitable name for the rule, for instance **Allow\_HTTP\_To\_DMZ**.
3. Now enter:
  - **Action:** Allow
  - **Source Interface:** any
  - **Source Network:** all-nets
  - **Destination Interface:** core
  - **Destination Network:** wwwsrv\_pub
4. Under the **Service** tab, select **http** in the **Pre-defined** dropdown list.
5. Click **OK**.

## 7.2.3. All-to-One Mappings (N:1)

CorePlus can be used to translate ranges and/or groups into just one IP address.

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	194.1.2.16-194.1.2.20, 194.1.2.30	http SETDEST all-to-one 192.168.0.50 80

This rule produces a N:1 translation of all addresses in the group (the range 194.1.2.16 - 194.1.2.20 and 194.1.2.30) to the IP 192.168.0.50.

- Attempts to communicate with 194.1.2.16, port 80, will result in a connection to 192.168.0.50
- Attempts to communicate with 194.1.2.30, port 80, will result in a connection to 192.168.0.50



### *Note*

*When 0.0.0.0/0 is the destination, All-to-One mapping is always done.*

## 7.2.4. Port Translation

Port Translation, also known as PAT (Port Address Translation), can be used to modify the source

or destination port.

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	wwwsrv_pub	TCP 80-85 SETDEST 192.168.0.50 1080

This rule produces a 1:1 translation of all ports in the range 80 - 85 to the range 1080 - 1085.

- Attempts to communicate with the web servers public address, port 80, will result in a connection to the web servers private address, port 1080.
- Attempts to communicate with the web servers public address, port 84, will result in a connection to the web servers private address, port 1084.



### Note

*In order to create a SAT Rule that allows port translation, a Custom Service must be used with the SAT Rule.*

## 7.2.5. Which Protocols can SAT handle?

Generally, static address translation can handle all protocols that allow address translation to take place. However, there are protocols that can only be translated in special cases, and other protocols that simply cannot be translated at all.

Protocols that are impossible to translate using SAT are most likely also impossible to translate using NAT. Reasons for this include:

- The protocol cryptographically requires that the addresses are unaltered; this applies to many VPN protocols.
- The protocol embeds its IP addresses inside the TCP or UDP level data, and subsequently requires that, in some way or another, the addresses visible on IP level are the same as those embedded in the data. Examples of this include FTP and logons to NT domains via NetBIOS.
- Either party is attempting to open new dynamic connections to the addresses visible to that party. In some cases, this can be resolved by modifying the application or the gateway configuration.

There is no definitive list of what protocols that can or cannot be address translated. A general rule is that VPN protocols cannot usually be translated. In addition, protocols that open secondary connections in addition to the initial connection can be difficult to translate.

Some protocols that are difficult to address translate may be handled by specially written algorithms designed to read and/or alter application data. These are commonly referred to as *Application Layer Gateways* or *Application Layer Filters*. CorePlus supports a number of such Application Layer Gateways and for more information please see Section 6.2, “Application Layer Gateways”.

## 7.2.6. Which SAT Rule is executed if several are matching?

CorePlus does not terminate the rule-set lookup upon finding a matching SAT rule. Instead, it continues to search for a matching Allow, NAT or FwdFast rule. Only when it has found such a matching rule does the gateway execute the static address translation.

Despite this, the first matching SAT rule found for each address is the one that will be carried out.

"Each address" above means that two SAT rules can be in effect at the same time on the same connection, provided that one is translating the sender address whilst the other is translating the destination address.

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	wwwsrv_pub	TCP 80-85 SETDEST 192.168.0.50 1080
2	SAT	lan	lannet	all-nets	Standard	SETSRC pubnet

The two above rules may both be carried out concurrently on the same connection. In this instance, internal sender addresses will be translated to addresses in the "pubnet" in a 1:1 relation. In addition, if anyone tries to connect to the public address of the web server, the destination address will be changed to its private address.

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	lan	lannet	wwwsrv_pub	TCP 80-85	SETDEST intrasrv 1080
2	SAT	any	all-nets	wwwsrv_pub	TCP 80-85	SETDEST wwwsrv-priv 1080

In this instance, both rules are set to translate the destination address, meaning that only one of them will be carried out. If an attempt is made internally to communicate with the web servers public address, it will instead be redirected to an intranet server. If any other attempt is made to communicate with the web servers public address, it will be redirected to the private address of the publicly accessible web server.

Again, note that the above rules require a matching Allow rule at a later point in the rule-set in order to work.

## 7.2.7. SAT and FwdFast Rules

It is possible to employ static address translation in conjunction with FwdFast rules, although return traffic must be explicitly granted and translated.

The following rules make up a working example of static address translation using FwdFast rules to a web server located on an internal network:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	ip_wan	http SETDEST wwwsrv 80
2	SAT	lan	wwwsrv	any	all-nets	80 -> All SETSRC ip_wan 80
3	FwdFast	any	all-nets	core	ip_wan	http
4	FwdFast	lan	wwwsrv	any	all-nets	80 -> All

We add a NAT rule to allow connections from the internal network to the Internet:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
5	NAT	lan	lannet	any	all-nets	All

What happens now?

- External traffic to ip\_wan:80 will match rules 1 and 3, and will be sent to wwwsrv. Correct.
- Return traffic from wwwsrv:80 will match rules 2 and 4, and will appear to be sent from ip\_wan:80. Correct.
- Internal traffic to ip\_wan:80 will match rules 1 and 3, and will be sent to wwwsrv. Almost correct; the packets will arrive at wwwsrv, but:
- Return traffic from wwwsrv:80 to internal machines will be sent directly to the machines themselves. This will not work, as the packets will be interpreted as coming from the wrong address.

We will now try moving the NAT rule between the SAT and FwdFast rules:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	ip_wan	http SETDEST wwwsrv 80
2	SAT	lan	wwwsrv	any	all-nets	80 -> All SETSRC ip_wan 80
3	NAT	lan	lannet	any	all-nets	All
4	FwdFast	any	all-nets	core	ip_wan	http
5	FwdFast	lan	wwwsrv	any	all-nets	80 -> All

What happens now?

- External traffic to ip\_wan:80 will match rules 1 and 4, and will be sent to wwwsrv. Correct.
- Return traffic from wwwsrv:80 will match rules 2 and 3. The replies will therefore be dynamically address translated. This changes the source port to a completely different port, which will not work.

The problem can be solved using the following rule-set:

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Parameters
1	SAT	any	all-nets	core	ip_wan	http SETDEST wwwsrv 80
2	SAT	lan	wwwsrv	any	all-nets	80 -> All SETSRC ip_wan 80
3	FwdFast	lan	wwwsrv	any	all-nets	80 -> All
4	NAT	lan	lannet	any	all-nets	All
5	FwdFast	lan	wwwsrv	any	all-nets	80 -> All

- External traffic to ip\_wan:80 will match rules 1 and 5, and will be sent to wwwsrv. Correct.
- Return traffic from wwwsrv:80 will match rules 2 and 3. Correct.
- Internal traffic to ip\_wan:80 will match rules 1 and 4, and will be sent to wwwsrv. The sender address will be the Clavister Security Gateway's internal IP address, guaranteeing that return traffic passes through the gateway.
- Return traffic will automatically be handled by the Clavister Security Gateway's stateful inspection mechanism.



---

# Chapter 8. User Authentication

This chapter describes how CorePlus implements user authentication.

- Overview, page 178
- Authentication Components, page 180
- Authentication Process, page 182

## 8.1. Overview

Before any user service request is authorized by gateway's security policies, CorePlus needs to verify the identity of that user through a process of authentication.

### Authentication Methods

Generally, the authentication process prompts the user to prove their identity. What the user supplies as proof can be:

- A. Something the user is. Unique attributes that are different for every person, such as a fingerprint.
- B. Something the user has, such as X.507 Digital Certificates, Passcard, or Public and Private Keys.
- C. Something the user knows such as a password.

Method A. requires some special devices to scan and read the feature presented, which is often expensive. Another problem is that the feature usually can't be replaced if it's lost. Methods B. and C. are therefore the most common in network security. However these also can have drawbacks. Keys, for example, might be intercepted, cards might be stolen, people might choose weak passwords that are easily guessed, or they may be simply bad at keeping a secret. B. and C. are therefore often combined. An example of this is a passcard that requires a password or pincode for use.

User authentication is frequently used in services, such as HTTP, FTP, and VPN. CorePlus uses a Username/Password combination as the primary authentication method, strengthened by encryption algorithms. More advanced and secure means of authentication include Public-Private Keys, X.509 Certificates, IPsec\IKE, IKE XAuth, and ID Lists.

### What makes a "good" Password?

To penetrate certain system and obtain the user or administrator's privileges, passwords are often subject to attacks by guesswork or systematic searches. To counter attacks, a password should:

- Be more than 8 characters with no repeats
- Use random characters not commonly used in phrases
- Contain lower and upper case characters
- Contain numbers and special characters

Passwords should also:

- Not be recorded anywhere
- Never revealed to anyone

- Changed regularly

Good passwords help secure networks including Layer 2 tunnels which use passwords for encryption.

## User Types

CorePlus has authentication schemes which support diverse users. These can be:

- Administrators
- Normal users accessing the network
- PPPoE/PPTP/L2TP users using PPP authentication methods
- IPsec/IKE users - the entities authentication during the IKE negotiation phases (Implemented by Pre-shared Keys or Certificates).
- IKE XAuth users - an extension to IKE authentication, occurring between negotiation phase 1 and phase 2
- user groups - groups of users that are subject to same criteria



## 8.2. Authentication Components

CorePlus can either use a locally stored database, or a database on an external server to provide user authentication.

### The Local User Database (UserDB)

The Local User Database is a built-in registry inside CorePlus which contains the profiles of authorized users and user groups. Users' names and passwords can be configured into this database, and the users having same privileges can be grouped together for make administration easier.

A user can be stored as a member into more than one group and any change made to the group propagates to each group member. Passwords are stored in the configuration using reversible cryptography. This is in order to be compatible with various challenge-response authentication methods such as CHAP. When the local user database is enabled, CorePlus consults its internal user profiles to authenticate the user before approving any user's request.

### External Authentication Servers

In a larger network topology, it is preferable to have one central database within a dedicated server or a cluster of servers to handle all the authentication information. When there is more than one Clavister Security Gateway in the network and thousands of users added or removed constantly, the administrator will not have to configure and maintain separate databases of authorized user profiles on each gateway. Instead, the external server can validate the username/password against its central database, which is easily administered. CorePlus supports the Remote Authentication Dial-in User Service (RADIUS) for external database authentication.

CorePlus, acting as a RADIUS client, sends user credentials and connection parameter information in the form of a RADIUS message to a RADIUS server. The server authenticates and authorizes the request, and sends back a RADIUS message in response. RADIUS authentication messages are sent as UDP messages via UDP port 1812. One or more external servers can be defined in the gateway to improve the availability of the RADIUS system.

To provide security for RADIUS messages, a common *shared secret* is configured on both the RADIUS client and the server. This shared secret enables basic encryption of the user's password when the RADIUS message is transmitted from the RADIUS client to the server, and is commonly configured as a relatively long text string. It can contain up to 100 characters and is case sensitive.

RADIUS uses PPP to transfer a username/password request between client and RADIUS server as well as using PPP authentication schemes such as PAP and CHAP.

### Authentication Agents

Four different agents built into CorePlus can be used to perform username/password authentication. They are:

- HTTP - Authentication via web browsing. Users surf to the gateway and login either through an HTML form or a "401 - Authentication Required" dialog.
- HTTPS - Authentication via secure web browsing. Similar to HTTP agent except that Host and Root Certificates are used to establish SSL connection to the gateway.
- XAUTH - Authentication during IKE negotiation in IPsec VPN (if the IPsec tunnel has been configured to require XAUTH authentication).
- PPP - Authentication when PPTP/L2TP tunnels are set up (if the PPTP/L2TP tunnel has been configured to require user authentication).

## Authentication Rules

A user authentication rule specifies:

- From where (i.e. receiving interface, source network) users are allowed to authenticate themselves at the gateway.
- Which agent will be used by CorePlus to prompt users for authentication.
- Where is the location of the database that CorePlus. consults to perform authentication. Is it the local database or from an external server?
- Timeouts that logout authenticated users automatically.



### **Note**

*When using XAUTH agent, there is no need to specify the receiving interface, or source network, as this information is not available at the XAUTH phase. For the same reason, only one XAUTH user authentication rule can be defined. XAUTH is only used to set up IPsec VPN tunnels.*

## 8.3. Authentication Process

CorePlus performs user authentication as follows:

- A user connects to the gateway to initiate authentication.
- CorePlus receives the user's request from an interface, and notes in the IP rule-set that this traffic is allowed to reach its core authentication agent.
- According to the authentication agent specified in the authentication rule, CorePlus prompts the user with an authentication request.
- The user replies by entering their identification information, a username/password pair.
- CorePlus validates the information with respect to the authentication source specified in the authentication rule, either the local database or an external database in a RADIUS server will be taken.
- If a matching entry in the database is found, the CorePlus responds with an approval message, otherwise rejection.
- CorePlus then forwards the approved user's further service requests to their desired destinations, if the service is allowed by an IP rule explicitly and the user is a member of the user(s)/group(s) defined on the address object of that rule. Requests from those failing the authentication step are discarded.
- After a certain time period, the authenticated user will be automatically logged out according to the timeout restrictions defined in the authentication rule.

### Example 8.1. Creating an authentication user group

In the example of an authentication address object in the Address Book, a user group "users" is used to enable user authentication on "lannet". This example shows how to configure the user group in the CorePlus database.

#### *Clavister FineTune*

##### Step A

1. Select the **Local Objects > User Databases** section of the target system in the tree view of the Security Editor.
2. Choose **New User Database...** from the context menu, the **User Database Properties** dialog box will be displayed.
3. Now enter:
  - **Name:** lannet\_auth\_users
  - **Comments:** folder for "lannet" authentication user group - "users"
4. Click the **OK**.

##### Step B

1. Select the **Local Objects > User Databases > lannet\_auth\_users** section of the target system in the tree view of the Security Editor.
2. Choose **New User...** from the context menu, the **User Properties** dialog box will be displayed.
3. Now enter:
  - **Username:** Enter the user's account name here, e.g. "user1"
  - **Password:** Enter the user's password
  - **Confirm Password:** Repeat the password
  - **Groups:** One user can be specified into more than one group. Enter the group names here separated by comma, e.g. "users" for this example.
4. Click the **OK**.
5. Repeat Step B to add all the "lannet" users having the membership of "users" group into the **lannet\_auth\_users** folder.



#### **Note**

*There are two default user groups, the administrators group and the auditors group. Users that are members of the administrators group are allowed to change the CorePlus configuration, while users that belong to the auditors group are only allowed to view the configuration. Press the buttons under the **Groups** edit box to grant these group memberships to a user.*

---

# Chapter 9. Virtual Private Networks

This chapter describes VPN usage with CorePlus.

- VPN overview, page 184
- IPsec, page 186
- IPsec tunnels, page 199
- PPTP/L2TP, page 214

## 9.1. VPN overview

### 9.1.1. The need for VPNs

Most networks today are connected to each other by the Internet. Business increasingly utilizes the Internet since it offers efficient and inexpensive communication. Issues of protecting local networks from Internet-based intrusion are being solved by gateways, intrusion detection systems and other security investments.

Private as well as corporate communication requires a means for data to travel across the Internet to its intended recipient without another party being able to read or alter it. It is equally important that the recipient can verify that no one is falsifying information, i.e. pretending to be someone else. VPNs meet this need, providing a highly cost efficient means of establishing secure links to parties that one wishes to exchange information with in a secure manner.

### 9.1.2. The basics of VPN Encryption

Cryptography provides the means to create VPNs across the Internet with no additional investments in connectivity. Cryptography is an umbrella expression covering 3 techniques and benefits:

<b>Confidentiality</b>	No one but the intended recipients is able to receive and understand the communication. Confidentiality is accomplished by encryption.
<b>Authentication and Integrity</b>	Proof for the recipient that the communication was actually sent by the expected sender, and that the data has not been modified in transit. This is accomplished by authentication, often by use of cryptographic keyed hashes.
<b>Non-repudiation</b>	Proof that the sender actually sent the data; the sender cannot later deny having sent it. Non-repudiation is usually a side-effect of authentication.

VPNs are normally only concerned with confidentiality and authentication. Non-repudiation is normally not handled at the network level but rather on a transaction (document-by-document) basis.

### 9.1.3. Planning a VPN

An attacker wishing to make use of a VPN connection will typically not attempt to crack the VPN encryption since this requires enormous work. Rather, they will see VPN traffic as an indication that there is something worth targeting on the other end of the connection. Typically, mobile clients and branch offices are far more attractive targets than the main corporate networks. Once inside those, getting to the corporate network becomes a much easier task.

In designing a VPN, there are many non-obvious issues that need to be addressed. This includes:

- Protecting mobile and home computers
- Restricting access through the VPN to needed services only, since mobile computers are vulnerable
- Creating DMZs for services that need to be shared with other companies through VPNs
- Adapting VPN access policies for different groups of users
- Creating key distribution policies

A common misconception is that VPN-connections are equivalents to the internal network from a security standpoint and that they can be connected directly to it with no further precautions. It is important to remember that although the VPN-connection itself may be secure, the total level of security is only as high as the security of the tunnel endpoints.

It is becoming increasingly common for users on the move to connect directly to their company's network via VPN from their laptops. However, the laptop itself is often not protected. In other words, an intruder can gain access to the protected network through an unprotected laptop and already-opened VPN connections.

A VPN connection should never be regarded as an integral part of a protected network. The VPN gateway should instead be located in a special DMZ or outside a gateway dedicated to this task. By doing this, you can restrict which services can be accessed via VPN and modem and ensure that these services are well protected against intruders. In instances where the gateway features an integrated VPN feature, it is usually possible to dictate the types of communication permitted. The CorePlus VPN module features such a facility.

### 9.1.3.1. Key Distribution

Key distribution schemes are best planned ahead of time. Issues that need to be addressed include:

- How will keys be distributed? Email is not a good idea. Phone conversations might be secure enough.
- How many different keys should be used? One key per user? One key per group of users? One key per LAN-to-LAN connection? One key for all users and one key for all LAN-to-LAN connections? You are probably better off using more keys than you think necessary today, since it becomes easier to adjust access per user (group) in the future.
- Should the keys be changed? If so, how often? In cases where keys are shared by multiple users, you may want to consider overlapping schemes, so that the old keys work for a short period of time when new keys have been issued.
- What happens when an employee in possession of a key leaves the company? If several users are using the same key, it should be changed.
- In cases where the key is not directly programmed into a network unit, such as a VPN gateway, how should the key be stored? On a floppy? As a pass phrase to memorize? On a smart card? If it is a physical token, how should it be handled?

## 9.2. IPsec

### 9.2.1. IPsec Basics

#### 9.2.1.1. Introduction to IPsec

IPsec, Internet Protocol Security, is a set of protocols defined by the IETF, Internet Engineering Task Force, to provide IP security at the network layer. An IPsec based VPN is made up by two parts:

- Internet Key Exchange protocol (IKE)
- IPsec protocols (AH/ESP/both)

The first part, IKE, is the initial negotiation phase, where the two VPN endpoints agree on which methods will be used to provide security for the underlying IP traffic. Furthermore, IKE is used to manage connections, by defining a set of Security Associations, SAs, for each connection. SAs are unidirectional, so there are usually at least two for each IPsec connection.

The second part is the actual IP data being transferred, using the encryption and authentication methods agreed upon in the IKE negotiation. This can be accomplished in a number of ways; by using IPsec protocols ESP, AH, or a combination of both.

The flow of events can be briefly described as follows:

- IKE negotiates how IKE should be protected
- IKE negotiates how IPsec should be protected
- IPsec moves data in the VPN

The following sections will describe each of these steps in detail.

#### 9.2.1.2. IKE, Internet Key Exchange

This section describes IKE, the Internet Key Exchange protocol, and the parameters that are used with it.

Encrypting and authenticating data is fairly straightforward, the only things needed are encryption and authentication algorithms, and the keys used with them. The Internet Key Exchange (IKE) protocol, IKE, is used as a method of distributing these "session keys", as well as providing a way for the VPN endpoints to agree on how the data should be protected.

IKE has three main tasks:

- Provide a means for the endpoints to authenticate each other
- Establish new IPsec connections (create SA pairs)
- Manage existing connections

IKE keeps track of connections by assigning a set of Security Associations, SAs, to each connection. An SA describes all parameters associated with a particular connection, such as the IPsec protocol used (ESP/AH/both) as well as the session keys used to encrypt/decrypt and/or authenticate/verify the transmitted data. An SA is, by nature, unidirectional, thus the need for more than one SA per connection. In most cases, where only one of ESP or AH is used, two SAs will be created for each connection, one describing the incoming traffic, and the other the outgoing. In cases where ESP and

AH are used in conjunction, four SAs will be created.

## IKE Negotiation

The process of negotiating session parameters consists of a number of phases and modes. These are described in detail in the below sections.

The flow of events can be summarized as follows:

### IKE Phase-1

- Negotiate how IKE should be protected

### IKE Phase-2

- Negotiate how IPsec should be protected
- Derive some fresh keying material from the key exchange in phase-1, to provide session keys to be used in the encryption and authentication of the VPN data flow

Both the IKE and the IPsec connections have limited lifetimes, described both in terms of time (seconds), and data (kilobytes). These lifetimes prevent a connection from being used too long, which is desirable from a cryptanalysis perspective.

The IPsec lifetime is generally shorter than the IKE lifetime. This allows for the IPsec connection to be re-keyed simply by performing another phase-2 negotiation. There is no need to do another phase-1 negotiation until the IKE lifetime has expired.

## IKE Proposals

An IKE proposal is a suggestion of how to protect data. The VPN device initiating an IPsec connection, the initiator, will send a list of proposals, a proposal-list, suggesting different methods of how to protect the connection.

The connection being negotiated can be either an IPsec connection protecting the data flow through the VPN, or it can be an IKE connection, protecting the IKE negotiation itself.

The responding VPN device, upon receiving this proposal-list, will choose the most suitable proposal according to its own security policy, and respond by specifying which one of the proposals it has chosen.

If no acceptable proposal can be found, it will respond by saying that no proposal could be accepted, and possibly provide a reason why.

The proposals contain all parameters needed, such as algorithms used to encrypt and authenticate the data, and other parameters as described in section IKE Parameters.

## IKE Phase-1 - IKE Security Negotiation

An IKE negotiation is performed in two phases. The first phase, phase-1, is used to authenticate the two VPN gateways or VPN Clients to each other, by confirming that the remote device has a matching Pre-Shared Key.

However since we do not want to publish too much of the negotiation in plaintext, we first agree upon a way of protecting the rest of the IKE negotiation. This is done, as described in the previous section, by the initiator sending a proposal-list to the responder. When this has been done, and the responder accepted one of the proposals, we try to authenticate the other end of the VPN to make sure it is who we think it is, as well as proving to the remote device; that we are who we claim to be. A technique known as a *Diffie Hellman Key Exchange* is used to initially agree a shared secret between the two parties in the negotiation and to derive keys for encryption.

Authentication can be accomplished through Pre-Shared Keys, certificates or public key encryption.



Pre-Shared Keys is the most common authentication method today. PSK and certificates are supported by the CorePlus VPN module.

## IKE Phase-2 - IPsec Security Negotiation

In phase two, another negotiation is performed, detailing the parameters for the IPsec connection.

In phase-2 we will also extract new keying material from the Diffie-Hellman key exchange in phase-1, to provide session keys to use in protecting the VPN data flow.

If PFS, Perfect Forwarding Secrecy, is used, a new Diffie-Hellman exchange is performed for each phase-2 negotiation. While this is slower, it makes sure that no keys are dependent on any other previously used keys; no keys are extracted from the same initial keying material. This is to make sure that, in the unlikely event that some key was compromised, no subsequent keys can be derived.

Once the phase-2 negotiation is finished, the VPN connection is established and ready for use.

## IKE Parameters

There are a number of parameters used in the negotiation process.

Below is a summary of the configuration parameters needed to establish a VPN connection. Understanding what these parameters do before attempting to configure the VPN endpoints is highly recommended, since it is of great importance that both endpoints are able to agree on all of these parameters.

When installing two Clavister Security Gateways as VPN endpoints, this process is reduced to comparing fields in two identical dialog boxes. However, it is not quite as easy when equipment from different vendors is involved.

### Endpoint Identification

This is a piece of data representing the identity of the VPN gateway. What this is exactly, depends on the authentication method used. When Pre-Shared Keys are used, this is a piece of data, generally a hex-string or some kind of "pass phrase", identifying this VPN gateway. The remote device has to have the same PSK in order for the VPN gateways to authenticate each other.

Authentication using Pre-Shared Keys is based on the Diffie-Hellman algorithm.

### Local and Remote Networks/ Hosts

These are the subnets or hosts between which IP traffic will be protected by the VPN. In a LAN-to-LAN connection, these will be the network addresses of the respective LANs.

If roaming clients are used, the remote network will most likely be set to 0.0.0.0/0, meaning that the roaming client may connect from anywhere.


### Tunnel / Transport Mode

IPsec can be used in two modes, tunnel or transport.

Tunnel mode indicates that the traffic will be tunneled to a remote device, which will decrypt/authenticate the data, extract it from its tunnel and pass it on to its final destination. This way, an eavesdropper will only see encrypted traffic going from one of VPN endpoint to another.

In transport mode, the traffic will not be tunneled, and is hence not applicable to VPN tunnels. It can be used to secure a connection from a VPN client directly to the Clavister Security Gateway, e.g. for IPsec protected remote configuration.

---

	<p>This setting will typically be set to "tunnel" in most configurations.</p>
<b>Remote Gateway</b>	<p>The remote gateway will be doing the decryption/authentication and pass the data on to its final destination. This field can also be set to "none", forcing the Clavister VPN to treat the remote address as the remote gateway. This is particularly useful in cases of roaming access, where the IP addresses of the remote VPN clients are not known beforehand. Setting this to "none" will allow anyone coming from an IP address conforming to the "remote network" address discussed above to open a VPN connection, provided they can authenticate properly.</p> <p>The remote gateway is not used in transport mode.</p>
<b>Main/Aggressive Mode</b>	<p>The IKE negotiation has two modes of operation, main mode and aggressive mode.</p> <p>The difference between these two is that aggressive mode will pass more information in fewer packets, with the benefit of slightly faster connection establishment, at the cost of transmitting the identities of the security gateways in the clear.</p> <p>When using aggressive mode, some configuration parameters, such as Diffie-Hellman groups, and PFS, can not be negotiated, resulting in a greater importance of having "compatible" configurations on both ends.</p>
<b>IPsec Protocols</b>	<p>The IPsec protocols describe how the data will be processed. The two protocols to choose from are AH, Authentication Header, and ESP, Encapsulating Security Payload.</p> <p>ESP provides encryption, authentication, or both. However, we do not recommend using encryption only, since it will dramatically decrease security.</p> <p>More on ESP in ESP (Encapsulating Security Payload).</p> <p>AH only provides authentication. The difference from ESP with authentication only is that AH also authenticates parts of the outer IP header, for instance source and destination addresses, making certain that the packet really came from who the IP header claims it is from.</p> <p>More on AH in AH (Authentication Header).</p>
	<p> <b>Note</b> <i>Clavister Security Gateways do not support AH.</i></p>
<b>IKE Encryption</b>	<p>This specifies the encryption algorithm used in the IKE negotiation, and depending on the algorithm, the size of the encryption key used.</p> <p>The algorithms supported by CorePlus IPsec are:</p> <ul style="list-style-type: none"><li>• AES</li><li>• Blowfish</li></ul>

---

- Twofish
- Cast128
- 3DES
- DES

DES is only included to be interoperable with other older VPN implementations. Use of DES should be avoided whenever possible, since it is an old algorithm that is no longer considered secure.

**IKE Authentication**

This specifies the authentication algorithms used in the IKE negotiation phase.

The algorithms supported by CorePlus IPsec are:

- SHA1
- MD5

**IKE DH (Diffie-Hellman) Group**

This specifies the Diffie-Hellman group to use when doing key exchanges in IKE.

The Diffie-Hellman groups supported by Clavister Security Gateway VPNs are:

- DH group 1 (768-bit)
- DH group 2 (1024-bit)
- DH group 5 (1536-bit)

The security of the key exchanges increase as the DH groups grow larger, as does the time of the exchanges.

**IKE Lifetime**

This is the lifetime of the IKE connection.

It is specified in time (seconds) as well as data amount (kilobytes). Whenever one of these expires, a new phase-1 exchange will be performed. If no data was transmitted in the last "incarnation" of the IKE connection, no new connection will be made until someone wants to use the VPN connection again.

**PFS**

With PFS disabled, initial keying material is "created" during the key exchange in phase-1 of the IKE negotiation. In phase-2 of the IKE negotiation, encryption and authentication session keys will be extracted from this initial keying material. By using PFS, Perfect Forwarding Secrecy, completely new keying material will always be created upon re-key. Should one key be compromised, no other key can be derived using that information.

PFS can be used in two modes, the first is PFS on keys, where a new key exchange will be performed in every phase-2 negotiation. The other type is PFS on identities, where the identities are also protected, by deleting the phase-1 SA every time a phase-2 negotiation has been finished, making sure no more than one phase-2 negotiation is encrypted using the same key.

PFS is generally not needed, since it is very unlikely that any

	encryption or authentication keys will be compromised.
<b>IPsec DH Group</b>	This is a Diffie-Hellman group much like the one for IKE. However, this one is used solely for PFS.
<b>IPsec Encryption</b>	<p>The encryption algorithm to use on the protected traffic.</p> <p>This is not needed when AH is used, or when ESP is used without encryption.</p> <p>The algorithms supported by Clavister Security Gateway VPNs are:</p> <ul style="list-style-type: none"> <li>• AES</li> <li>• Blowfish</li> <li>• Twofish</li> <li>• Cast128</li> <li>• 3DES</li> <li>• DES</li> </ul>
<b>IPsec Authentication</b>	<p>This specifies the authentication algorithm used on the protected traffic.</p> <p>This is not used when ESP is used without authentication, although it is not recommended to use ESP without authentication.</p> <p>The algorithms supported by Clavister Security Gateway VPNs are:</p> <ul style="list-style-type: none"> <li>• SHA1</li> <li>• MD5</li> </ul>
<b>IPsec Lifetime</b>	This is the lifetime of the VPN connection. It is specified in both time (seconds) and data amount (kilobytes). Whenever either of these values is exceeded, a re-key will be initiated, providing new IPsec encryption and authentication session keys. If the VPN connection has not been used during the last re-key period, the connection will be terminated, and re-opened from scratch when the connection is needed again.

### 9.2.1.3. IKE Authentication Methods (Manual, PSK, certificates)

#### Manual Keying

The "simplest" way of configuring a VPN is by using a method called "manual keying". This is a method where IKE is not used at all; the encryption and authentication keys as well as some other parameters are directly configured on both sides of the VPN tunnel.



**Note**

*Clavister Security Gateways do not support Manual Keying.*

#### Advantages

Since it is very straightforward it will be quite interoperable. Most interoperability problems encountered today are in IKE. Manual keying completely bypasses IKE and sets up its own set of IPsec SAs.

### Disadvantages

It is an old method, which was used before IKE came into use, and is thus lacking all the functionality of IKE. This method therefore has a number of limitations, such as having to use the same encryption/authentication key always, no anti-replay services, and it is not very flexible. There is also no way of assuring that the remote host/gateway really is the one it says it is.

This type of connection is also vulnerable for something called "replay attacks", meaning a malicious entity which has access to the encrypted traffic can record some packets, store them, and send them to its destination at a later time. The destination VPN endpoint will have no way of telling if this packet is a "replayed" packet or not. Using IKE eliminates this vulnerability.

### Pre-Shared Keys

Using a Pre-shared Key (PSK) is a method where the endpoints of the VPN "share" a secret key. This is a service provided by IKE, and thus has all the advantages that come with it, making it far more flexible than manual keying.

### Advantages

Pre-Shared Keying has a lot of advantages over manual keying. These include endpoint authentication, which is what the PSKs are really for. It also includes all the benefits of using IKE. Instead of using a fixed set of encryption keys, session keys will be used for a limited period of time, where after a new set of session keys are used.

### Disadvantages

One thing that has to be considered when using Pre-Shared Keys is key distribution. How are the Pre-Shared Keys distributed to remote VPN clients and gateways? This is a major issue, since the security of a PSK system is based on the PSKs being secret. Should one PSK be compromised, the configuration will need to be changed to use a new PSK.

### Certificates

Each VPN gateway has its own certificate, and one or more trusted root certificates.

The authentication is based on several things:

- That each endpoint has the private key corresponding to the public key found in its certificate, and that nobody else has access to the private key.
- That the certificate has been signed by someone that the remote gateway trusts.

### Advantages

Added flexibility. Many VPN clients, for instance, can be managed without having the same pre-shared key configured on all of them, which is often the case when using pre-shared keys and roaming clients. Instead, should a client be compromised, the client's certificate can simply be revoked. No need to reconfigure every client.

### Disadvantages

Added complexity. Certificate-based authentication may be used as part of a larger public key infrastructure, making all VPN clients and gateways dependent on third parties. In other words, there are more things that have to be configured, and there are more things that can go wrong.

### 9.2.1.4. IPsec Protocols (ESP/AH)

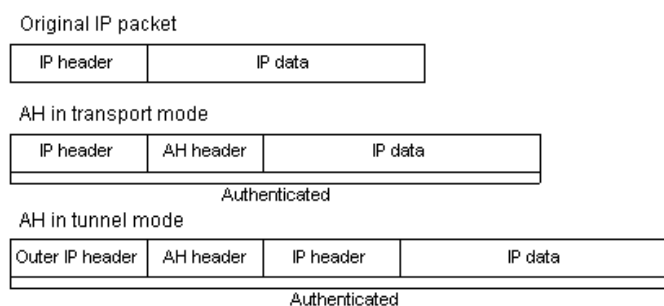
The IPsec protocols are the protocols used to protect the actual traffic being passed through the VPN. The actual protocols used and the keys used with those protocols are negotiated by IKE.

There are two protocols associated with IPsec, AH and ESP. These are covered in the sections below.

#### AH (Authentication Header)

AH is a protocol used for authenticating a data stream. It uses a cryptographic hash function to produce a MAC from the data in the IP packet. This MAC is then transmitted with the packet, allowing the remote gateway to verify the integrity of the original IP packet, making sure the data has not been tampered with on its way through the Internet.

**Figure 9.1. The AH protocol**



Apart from the IP packet data, AH also authenticates parts of the IP header.

The AH protocol inserts an AH header after the original IP header, and in tunnel mode, the AH header is inserted after the outer header, but before the original, inner, IP header.

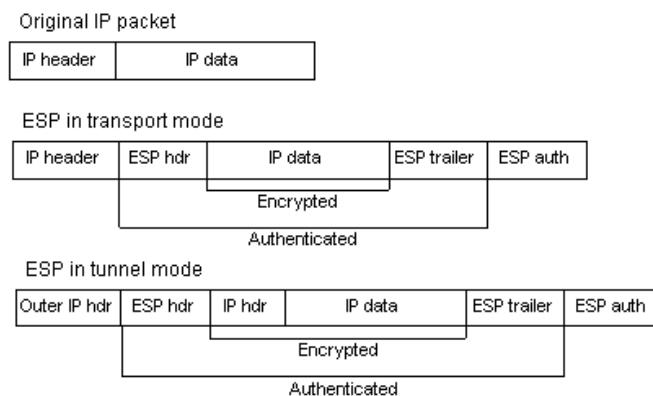
#### ESP (Encapsulating Security Payload)

The ESP protocol inserts an ESP header after the original IP header, in tunnel mode, the ESP header is inserted after the outer header, but before the original, inner, IP header.

All data after the ESP header is encrypted and/or authenticated. The difference from AH is that ESP also provides encryption of the IP packet. The authentication phase also differs in that ESP only authenticates the data after the ESP header; thus the outer IP header is left unprotected.

The ESP protocol is used for both encryption and authentication of the IP packet. It can also be used to do either encryption only, or authentication only.

**Figure 9.2. The ESP protocol**



### 9.2.1.5. NAT Traversal

Both IKE and IPsec protocols present a problem in the functioning of NAT. Both protocols were not designed to work through NATs and because of this, a technique called "NAT traversal" has evolved. NAT traversal is an add-on to the IKE and IPsec protocols that allows them to function when being NATed.

NAT traversal is divided into two parts:

- Additions to IKE that lets IPsec peers tell each other that they support NAT traversal, and the specific versions of the draft they support.
- Changes to the ESP encapsulation. If NAT traversal is used, ESP is encapsulated in UDP, which allows for more flexible NATing.

Below is a more detailed description of the changes made to the IKE and IPsec protocols.

NAT traversal is only used if both ends has support for it. For this purpose, NAT traversal aware VPNs send out a special "vendor ID", telling the other end that it understand NAT traversal, and which specific versions of the draft it supports.

NAT detection: Both IPsec peers send hashes of their own IP addresses along with the source UDP port used in the IKE negotiations. This information is used to see whether the IP address and source port each peer uses is the same as what the other peer sees. If the source address and port have not changed, then the traffic has not been NATed along the way, and NAT traversal is not necessary. If the source address and/or port has changed, then the traffic has been NATed, and NAT traversal is used.

Once the IPsec peers have decided that NAT traversal is necessary, the IKE negotiation is moved away from UDP port 500 to port 4500. This is necessary since certain NAT devices treat UDP packet to port 500 differently from other UDP packets in an effort to work around the NAT problems with IKE. The problem is that this special handling of IKE packets may in fact break the IKE negotiations, which is why the UDP port used by IKE has changed.

Another problem NAT traversal resolves is that the ESP protocol is an IP protocol. There is no port information like in TCP and UDP, which makes it impossible to have more than one NATed client connected to the same remote gateway and the same time. Because of this, ESP packets are encapsulated in UDP. The ESP-UDP traffic is sent on port 4500, the same port as IKE when NAT traversal is used. Once the port has been changed all following IKE communications are done over port 4500. Keepalive packets are also being sent periodically to keep the NAT mapping alive.

NAT traversal drafts supported by CorePlus:

- draft-ietf-ipsec-nat-t-ike-00

- draft-ietf-ipsec-nat-t-ike-01
- draft-ietf-ipsec-nat-t-ike-02
- draft-ietf-ipsec-nat-t-ike-03

### NAT Traversal Configuration

Most NAT traversal functionality is completely automatic and in the initiating gateway no special configuration is needed. However for responding gateways two points should be noted:

- On responding gateways, the Remote Gateway field is used as a filter on the source IP of received IKE packets. This should be set to allow the NATed IP address of the initiator.
- Individual pre-shared keys can not be used where multiple clients connecting to one remote gateway gets NATed out through the same address. Having the same pre-shared key on all clients will work. However, this is not recommended. The preferred way is to use certificates instead.

## 9.2.2. Proposal Lists

To agree on the VPN connection parameters, a negotiation process is performed. As the result of the negotiations, the IKE and IPsec security associations (SAs) are established. As the name implies, a proposal is the starting point for the negotiation. A proposal defines encryption parameters, for instance encryption algorithm, life times etc, that the VPN gateway supports.

There are two types of proposals, IKE proposals and IPsec proposals. IKE proposals are used during IKE Phase-1 (IKE Security Negotiation), while IPsec proposals are used during IKE Phase-2 (IPsec Security Negotiation).

A Proposal List is used to group several proposals. During the negotiation process, the proposals in the proposal list are offered to the remote VPN gateway one after another until a matching proposal is found. Several proposal lists can be defined in CorePlus for different VPN scenarios. Two IKE proposal lists and two IPsec proposal lists are defined by default in the Global Namespace.

The `ike-roamingclients` and `esp-tn-roamingclients` proposal lists are suitable for VPN tunnels that are used for roaming VPN clients. These proposal lists are compatible with the default proposal lists in the Clavister VPN Client.

As the name implies, the `ike-lantolan` and `esp-tn-lantolan` are suitable for LAN-to-LAN VPN solutions. These proposal lists are trimmed to include only AES and 3DES based proposals.

#### Example 9.1. Using a Proposal List

This example shows how to create and use an IPsec Proposal List for use in the VPN tunnel. It will propose 3DES and DES as encryption algorithms. The hash function SHA1 and MD5 will both be used in order to check if the data packet is altered while being transmitted. Note that this example does not illustrate how to add the specific IPsec tunnel object. It will also be used in a later example.

##### *Clavister FineTune*

First create a list of IPsec Algorithms:

1. Go to **Local Objects > VPN Settings > IPsec Proposal Lists**
2. Choose **New IPsec Proposal List** and the **IPsec Proposal List Properties** dialog box will appear
3. Enter a name for the list, in this case **esp-l2tptunnel**



4. Select **Transport** as the **Encapsulation Mode**

5. Now check the following:

- **DES**
- **3DES**
- **SHA1-96**
- **MD5-96**

6. For **Kilobytes** enter 250000

7. For **Seconds** enter 3600

8. Click **OK**

Now apply the proposal list to the IPSec tunnel:

1. Go to **Interfaces > IPSec Tunnels**

2. Right-click the target IPSec tunnel, choose **Properties** and the **IPSec Tunnel Properties** dialog will appear

3. Select **esp-I2tptunnel** as the **IPSec Proposal List**

4. Click **OK**

## 9.2.3. Pre-shared Keys

Pre-Shared Keys are used to authenticate VPN tunnels. The keys are secrets that are shared by the communicating parties before communication takes place. To communicate, both parties prove that they know the secret. The security of a shared secret depends on how "good" a passphrase is. Passphrases that are common words are for instance extremely vulnerable to dictionary attacks.

### Example 9.2. Using a Pre-Shared key

This example shows how to create a Pre-shared Key and apply it to a VPN tunnel. Since regular words and phrases are vulnerable to dictionary attacks, they should not be used as secrets. Here the pre-shared key is a randomly generated hexadecimal key. Note that this example does not illustrate how to add the specific IPSec tunnel object.

#### *Clavister FineTune*

First create a Pre-shared Key:

1. Go to **Local Objects > VPN Settings > Pre-Shared Keys**

2. Choose **New Pre-Shared Key** from the context menu and the **Pre-shared Key Properties** dialog will appear

3. Enter a name for the pre-shared key eg. MyPSK.

4. Choose **Hexadecimal Key** in the **Type** control

5. Click **Generate Random Key** to generate a key from the **Shared Secret** textbox

6. Click **OK**

Now apply the Pre-shared Key to the IPSec tunnel:

1. Go to **Interfaces > IPSec Tunnels**

2. Right-click on the target IPSec tunnel and choose **Properties** to display **IPSec Tunnel Properties**

3. In the **Authentication** tab, choose **Pre-Shared Key**

4. Select **MyPSK** in the **Pre-Shared key** dropdown list

5. Click **OK**

## 9.2.4. Identification Lists

When X.509 certificates are used as authentication method for IPsec tunnels, the Clavister Security Gateway will accept all remote gateways or VPN clients that are capable of presenting a certificate signed by any of the trusted Certificate Authorities. This can be a potential problem, especially when using roaming clients.

Consider the scenario of travelling employees being given access to the internal corporate networks using VPN clients. The organization administers their own Certificate Authority, and certificates have been issued to the employees. Different groups of employees are likely to have access to different parts of the internal networks. For instance, members of the sales force need access to servers running the order system, while technical engineers need access to technical databases.

Since the IP addresses of the travelling employees VPN clients cannot be known beforehand, the incoming VPN connections from the clients cannot be differentiated. This means that the gateway is unable to control the access to various parts of the internal networks.

The concept of Identification Lists presents a solution to this problem. An identification list contains one or more identities (IDs), where each identity corresponds to the subject field in an X.509 certificate. Identification lists can thus be used to regulate what X.509 certificates that are given access to what IPsec tunnels.

### Example 9.3. Using an Identity List

This example shows how to create and use an Identification List for use in the VPN tunnel. This Identification List will contain one ID with the type DN, distinguished name, as the primary identifier. Note that this example does not illustrate how to add the specific IPsec tunnel object.

#### *Clavister FineTune*

First create an Identification List:

1. Go to **Local Objects > VPN Settings > Identification Lists** for the target system
2. Choose **New Identification List** from the context menu and the **Create Identification List** dialog will appear
3. Enter a name for the identification list, e.g. MyIDList
4. Click **OK**

Then, create an ID:

1. Go to **Local Objects > VPN Settings > Identification Lists > MyIDList** for the target system
2. Choose **New ID** from the context menu and the **ID Properties** dialog will appear
3. Select **DN** in the **Type** control
4. Now enter:
  - **Common Name:** John Doe
  - **Organization Name:** Clavister
  - **Organizational Unit:** Support
  - **Country:** Sweden
  - **Email Address:** john.doe@Clavister.com
5. Click **OK**

Finally, apply the Identification List to the IPsec tunnel:

1. Go to **Interfaces > IPsec Tunnels**
2. Right-click on the target IPsec tunnel object and choose **Properties** and the **IPsec Tunnel Properties** dialog will appear
3. In the **Authentication** tab, choose **X.509 Certificate**
4. Select **MyIDList** as the **Identification List**
5. Click **OK**

## 9.3. IPsec tunnels

### 9.3.1. Overview of IPsec tunnels

An IPsec Tunnel defines an endpoint of an encrypted tunnel. Each IPsec Tunnel is interpreted as a logical interface by CorePlus, with the same filtering, traffic shaping and configuration capabilities as regular interfaces.

When another Clavister Security Gateway or Clavister VPN Client (or any IPsec compliant product) tries to establish a IPsec VPN tunnel to the Clavister Security Gateway, the configured IPsec Tunnels are evaluated. If a matching IPsec Tunnel definition is found, the IKE and IPsec negotiations then take place, resulting in a IPsec VPN tunnel being established.

Note that an established IPsec tunnel does *not* automatically mean that all traffic from that IPsec tunnel is trusted. On the contrary, network traffic that has been decrypted will be transferred to the ruleset for further evaluation. The source interface of the decrypted network traffic will be the name of the associated IPsec Tunnel. Furthermore, a Route or an Access rule, in the case of a roaming client, has to be defined to have the CorePlus accept certain source IP addresses from the IPsec tunnel.

For network traffic going in the opposite direction, that is, going into a IPsec tunnel, a reverse process takes place. First, the unencrypted traffic is evaluated by the ruleset. If a rule and route matches, CorePlus tries to find an established IPsec tunnel that matches the criteria. If not found, CorePlus will try to establish a tunnel to the remote gateway specified by the matching IPsec Tunnel definition.



#### **Note**

*IKE and ESP/AH traffic are sent to the IPsec engine before the ruleset is consulted. Encrypted traffic to the gateway therefore does not need to be allowed in the ruleset. This behaviour can be changed in the **IPsec Advanced Settings** section.*

### 9.3.2. LAN to LAN tunnels with a Pre-shared Key

A VPN can allow geographically distributed Local Area Networks (LANs) to communicate securely over the public internet. In a corporate context this means LANs at geographically separate sites can communicate with a level of security comparable to that existing if they communicated through a dedicated, private link.

Secure communication is achieved through the use of IPsec tunneling, with the tunnel extending from the VPN gateway at one location to the VPN gateway at another location. The Clavister Security Gateway is therefore the implementor of the VPN, while at the same time applying normal security surveillance of traffic passing through the tunnel. This section deals specifically with setting up Lan to Lan tunnels created with a Pre-shared Key (PSK).

A number of steps are required to set up LAN to LAN tunnels with PSK:

- If both local and remote gateways are Clavister Security Gateways, define the host and networks in the **Global Namespace**
- Set up a Pre-shared Key or secret for the VPN tunnel.
- Set up the **VPN tunnel properties**.
- Set up the **Route**(an **Access** entry isn't needed from V.8.2 onwards).
- Set up the **Rules** (2-way tunnel requires 2 rules).

**Example 9.4. Setup a LAN to LAN VPN tunnel with PSK****Clavister FineTune**

In this example, a LAN behind a local gateway called **Stockholm** will have a secure VPN tunnel defined to a LAN behind a remote gateway called **Helsinki**. The following are the steps to take on **Stockholm**. **Helsinki** must be configured to match these settings.

1. If both local and remote gateways are Clavister Security Gateways, go to **Edit > Move to namespace** to put **ip\_wan** and **lannet** into the **Global Namespace**
2. Go to **File > New Pre-Shared Key** in the **Pre-Shared Key** section to create the **PSK**. A hexadecimal key could be created with the **Generate Random Key** button.
3. In the **VPN Tunnel properties** section in the **Interface** folder, set up the tunnel properties:
  - **Name:** *HelsinkiVPN* (use this name in **Access** and **Rules** sections)
  - **Local Network:** Stockholm\_lannet
  - **Remote Network:** Helsinki\_lannet
  - **Remote Gateway:** Helsinki\_ip\_wan

Select the **IKE proposal** and **IPsec proposal** for the tunnel then, via the **Authentication** tab, select the **Pre-Shared Key** created previously.
4. In the **Routes** section in the **Routing** folder go to **File > New route**. **Helsinki\_lannet** is routed through the **HelsinkiVPN**.
5. Create an outgoing traffic rule for internal LAN traffic going to the LAN behind the **Helsinki** gateway. The parameters would be:
  - **Action:** Allow
  - **Source Interface:** lan
  - **Source Network:** Stockholm\_lannet
  - **Destination Interface:** HelsinkiVPN
  - **Destination Network:** Helsinki\_lannet
  - **Service:** All
6. Create a rule for incoming traffic from the remote network on the VPN interface with parameters:
  - **Action:** Allow
  - **Source Interface:** HelsinkiVPN
  - **Source Network:** Helsinki\_lannet
  - **Destination Interface:** any
  - **Destination Network:** Stockholm\_lannet
  - **Service:** All

**Note**

*In the example above, a **Destination Interface** value of any allows pinging of the internal IP of the gateway through the tunnel. A value of **lan** prevents this as it's routed on the **core** interface. A pre-defined service could be used instead of **All**.*

## 9.3.3. Roaming Clients

An employee who is on the move who needs to access a central corporate server from a notebook computer from different locations is a typical example of a roaming client. Apart from the need for

secure VPN access, the other major issue with roaming clients is that the mobile user's IP address is often not known beforehand. To handle the unknown IP address the CorePlus can dynamically add routes to the routing table as tunnels are established.

## Dealing with unknown IP addresses

If the IP address of the client is not known before hand then the Clavister Security Gateway needs to create a route in it's routing table dynamically as each client connects. In the example below this is the case and the IPsec tunnel is configured to dynamically add routes.

If clients are to be allowed to roam in from everywhere, irrspective of their IP address, then the **Remote Network** needs to be set to **all-nets** (IP address: 0.0.0.0/0) which will allow all existing IPv4-addresses to connect through the tunnel.

When configuring VPN tunnels for roaming clients it is usually not necessary to add to or modify the proposal lists that are pre-configured in CorePlus.

### 9.3.3.1. PSK based client tunnels

#### Example 9.5. Setting up a PSK based VPN tunnel for roaming clients

This example describes how to configure an IPsec tunnel at the head office Clavister Security Gateway for roaming clients that connect to the office to gain remote access. The head office network uses the 10.0.1.0/24 network span with external gateway IP ip\_wan.

##### *Clavister FineTune*

A. Create a pre-shared key for IPsec authentication:

1. Go to **Local Objects > VPN Settings > Pre-Shared Keys**
2. From the context menu select **New Pre-Shared key** and the **Pre-Shared Key Properties** dialog will appear.
3. Now enter:
  - **Name:** Enter a name for the pre-shared key eg."SecretKey"
  - **Shared Secret:** Enter a secret passphrase
  - **Confirm Secret:** Enter the secret passphrase again
4. Click **OK**.

B. Configure the IPsec tunnel:

1. Go to **Interfaces > IPsec Tunnels**
2. From the context menu select **New IPsec Tunnel** and the **IPsec Tunnel Properties** dialog will appear.
3. Now enter:
  - **Name:** RoamingIPsecTunnel
  - **Local Network:** 10.0.1.0/24 (the local network that the roaming users will connect to)
  - **Remote Network:** all-nets
  - **Remote Endpoint:** (None)
  - **Encapsulation Mode:** Tunnel
4. Select the **IKE Proposal List** for the tunnel.
5. Select the **IPsec Proposal List** for the tunnel.
6. In the **Authentication** tab select the pre-shared key created earlier.

7. Under the **Routing** tab enable the setting **Dynamically add route to the remote network when a tunnel is established**.
  8. Click **OK**.
- C. Finally configure the IP rule-set to allow traffic inside the tunnel.

### 9.3.3.2. Self-signed Certificate based client tunnels

#### Example 9.6. Setting up a Self-signed Certificate based VPN tunnel for roaming clients

This example describes how to configure an IPsec tunnel at the head office Clavister Security Gateway for roaming clients that connect to the office to gain remote access. The head office network uses the 10.0.1.0/24 network span with external gateway IP ip\_wan.

##### *Clavister FineTune*

A. Create a Self-signed Certificate for IPsec authentication:

1. Go to **Local Objects > Certificates**
2. From the context menu select **New Certificate** and the **Certificate Properties** dialog will appear.
3. Enter a descriptive **name**, for instance MyCertificate.
4. Make sure the **Create self-signed certificate** option is selected.
5. Proceed to the **Properties** tab.
6. Enter values for the required fields.
7. Make sure that the dates in **Certificate Validity** is not set too "low" since this makes the gateway certificate useless.
8. Click **OK**.

B. Import all the client's self-signed certificates:

1. Go to **Local Objects > Certificates**
2. From the context menu select **Import Certificate**.
3. Locate your exported client certificate and click **Open** and **OK**.
4. Continue to import all your client certificates according to the instruction in the previous step.
5. Finally export your gateway certificate, since that certificate is going to be imported by remote clients:
  - Go to **Local Objects > Certificates**
  - Select your gateway certificate in the grid and from the context menu, select **Export Certificate**
  - Pick a name and location for the certificate and click **Save**

C. Create Identification Lists:

1. Go to **Local Objects > VPN Settings > Identification Lists**
2. From the context menu select **New Identification List** and the **Identification List Properties** dialog will appear.
3. Enter a descriptive **name**, in this example *sales*.
4. Click **OK**.
5. Select your created identification list, right-click and select **New ID...** and the **ID Properties** dialog will appear.

6. Select **Email** as **Type**.
  7. In the **Value** field, enter the email address selected when you created the certificate on the client.
  8. Click **OK**.
  9. Create a new ID for every client that you want to grant access rights according to the instructions above.
- D. Configure the IPsec tunnel:
1. Go to **Interfaces > IPsec Tunnels**
  2. From the context menu select **New IPsec Tunnel** and the **IPsec Tunnel Properties** dialog will appear.
  3. Now enter:
    - **Name:** RoamingIPsecTunnel
    - **Local Network:** 10.0.1.0/24 (the local network that the roaming users will connect to)
    - **Remote Network:** all-nets
    - **Remote Endpoint:** (None)
    - **Encapsulation Mode:** Tunnel
  4. Select the **IKE Proposal List** for the tunnel.
  5. Select the **IPsec Proposal List** for the tunnel.
  6. In the **Authentication** tab choose X.509 Certificate as authentication method and select your newly created gateway certificate as **gateway Certificate**
  7. Continue to add all your client certificates in the **Root Certificate(s)** section.
  8. Select your ID List that you want to associate with your VPN Tunnel under the **Identification List** section. In our case that will be *sales*.
  9. Under the **Routing** tab enable the setting **Dynamically add route to the remote network when a tunnel is established**.
  10. Click **OK**.
- E. Finally configure the IP rule-set to allow traffic inside the tunnel.

### 9.3.3.3. CA Server issued Certificates based client tunnels

Setting up client tunnels using a Certification Authority issued X.509 certificate is largely the same as using Self-Signed certificates with the exception of a couple of steps. Most importantly, it is the responsibility of the administrator to acquire the appropriate certificate from an issuing authority. With some systems, such as Windows 2000 Server, there is built-in access to a CA server (in Windows 2000 Server this is found in **Certificate Services**). For more information on CA server issued certificates see Section 3.7, “X.509 Certificates”.

#### Example 9.7. Setting up a CA Server issued Certificate based VPN tunnel for roaming clients

This example describes how to configure an IPsec tunnel at the head office Clavister Security Gateway for roaming clients that connect to the office to gain remote access. The head office network uses the 10.0.1.0/24 network span with external gateway IP ip\_wan.

##### **Clavister FineTune**

A. Create a Self-signed Certificate for IPsec authentication:

1. Go to **Local Objects > Certificates**



2. From the context menu select **New Certificate** and the **Certificate Properties** dialog will appear.
3. Enter a descriptive **name**, for instance MyCertificate.
4. Make sure the **Create self-signed certificate** option is selected.
5. Proceed to the **Properties** tab.
6. Type in the corresponding fields on your own choosing.
7. Make sure that the dates in **Certificate Validity** is not set to "low" since this makes the gateway certificate useless.
8. Click **OK**.

B. Import CA servers root certificate:

1. Go to **Local Objects > Certificates**
2. From the context menu select **Import Certificate**.
3. Locate your CA root certificate and click **Open** and **OK**.
4. Finally export your gateway certificate, since that certificate is going to be imported by remote clients:
  - Go to **Local Objects > Certificates**
  - Select your gateway certificate in the grid and from the context menu, select **Export Certificate**
  - Pick a name and location for the certificate and click **Save**

C. Create Identification Lists:

1. Go to **Local Objects > VPN Settings > Identification Lists**
2. From the context menu select **New Identification List** and the **Identification List Properties** dialog will appear.
3. Enter a descriptive **name**, in this example *sales*.
4. Click **OK**.
5. Select your created identification list, right-click and select **New ID...** and the **ID Properties** dialog will appear.
6. Select **DN** as **Type**.
7. Type in the information entered earlier for the user certificate.
8. Click **OK**.
9. Create a new ID for every client that you want to grant access rights according to the instructions above.

D. Configure the IPsec tunnel:

1. Go to **Interfaces > IPsec Tunnels**
2. From the context menu select **New Ipsec Tunnel** and the **IPsec Tunnel Properties** dialog will appear.
3. Now enter:
  - **Name:** RoamingIPsecTunnel
  - **Local Network:** 10.0.1.0/24 (the local network that the roaming users will connect to)
  - **Remote Network:** all-nets
  - **Remote Endpoint:** (None)
  - **Encapsulation Mode:** Tunnel
4. Select the **IKE Proposal List** for the tunnel.
5. Select the **IPsec Proposal List** for the tunnel.

6. In the **Authentication** tab choose X.509 Certificate as authentication method and select your newly created gateway certificate as **gateway Certificate**
  7. Continue to add your CA server root certificate in the **Root Certificate(s)** section.
  8. Select your ID List that you want to associate with your VPN Tunnel under the **Identification List** section. In our case that will be *sales*.
  9. Under the **Routing** tab enable the setting **Dynamically add route to the remote network when a tunnel is established**.
  10. Click **OK**.
- E. Finally configure the IP rule-set to allow traffic inside the tunnel.

### 9.3.3.4. Using Config Mode

Config Mode is an extension to IKE that allows CorePlus to provide LAN configuration information to the remote VPN Client. If a tunnel is opened in this mode, the client is able to address individual servers on the remote network using a network name instead of an IP address.

With IKE Configuration Mode enabled for a tunnel, the gateway is capable of allocating this internal network information to the remote client who is requesting an IPsec connection to a LAN. In CorePlus the source of this information can be either static IP addresses specified in the Config-Mode Pool object or internally/externally configured DHCP servers. Information collected from an internal/external DHCP server is managed in CorePlus by an object called the IP pool system. This system maintains a pool or cache of IP addresses collected from a DHCP server and automatically renews leases when the lease time is about to expire. It also manages additional information such as DNS and WINS/NBNS as would an ordinary DHCP server.

## 9.3.4. Fetching CRLs from an alternate LDAP server

An X.509 root certificate usually includes the IP address or hostname of the Certificate Authority to contact when certificates or Certificate Revocation Lists need to be downloaded to the Clavister Security Gateway. Lightweight Directory Access Protocol (LDAP) is used for these downloads.

However, in some scenarios, this information is missing, or the administrator wishes to use another LDAP server. The LDAP configuration section can then be used to manually specify alternate LDAP servers.

### Example 9.8. Setting up an LDAP server

This example shows how to manually setup and specify a LDAP server.

#### *Clavister FineTune*

1. Go to **Local Objects > VPN Settings > LDAP**
2. Choose **New LDAP Server** from the context menu and the **LDAP Server Properties** dialog will appear
3. Now enter:
  - **IP Address:** 192.168.101.146
  - **Username:** myusername
  - **Password:** mypassword
  - **Port:** 389
4. Click **OK**

## 9.3.5. Troubleshooting with *ikesnoop*

### VPN Tunnel Negotiation

When setting up a VPN, problems can arise because the initial negotiation fails when the devices at either end of a VPN tunnel try to agree on which protocols and encryption methods they will use to communicate. The *ikesnoop* console command with the *verbose* option is a tool that can be used to identify the source of such problems by showing the details of this negotiation.

### Using *ikesnoop*

The command can be entered in one of three ways: Via the Clavister FineTune console, the RS232 Console or via the **fwctl** management tool. The output from *ikesnoop verbose* can be troublesome to interpret by an administrator seeing it for the first time. Presented below is a typical *ikesnoop verbose* output with annotations to explain it. The VPN tunnel negotiation considered is based on Pre-shared Keys, a negotiation based on Certificates is not discussed here but the principles are similar. (Complete *ikesnoop* command options can be found in Appendix B, *CLI Reference*).

### The Client and the Server

The two parties involved in the tunnel negotiation are referred to in this section as the *client* and server. In this context the word *client* is used to refer to the device which is the *initiator* of the VPN negotiation and the *server* refers to the device which is the *responder*.

### Step 1. Client Initiates Exchange By Sending Proposal

The *ikesnoop verbose* command output initially shows the proposal list that the client in the exchange first sends to the server detailing the protocols and encryption methods it can support. The purpose for the proposal is that the client is trying to find a matching set of protocols/methods supported by the server. The server examines the proposal and attempts to find a combination of the protocols/methods sent by the client which it supports.

```
IkeSnoop: Received IKE packet from 192.168.0.10:500 Exchange type :
          Identity Protection (main mode) ISAKMP Version : 1.0

Flags           :
Cookies        : 0x6098238b67d97ea6 -> 0x00000000
Message ID     : 0x00000000
Packet length  : 324 bytes
# payloads     : 8
Payloads:
  SA (Security Association)
    Payload data length : 152 bytes
    DOI : 1 (IPsec DOI)
      Proposal 1/1
        Protocol 1/1
          Protocol ID           : ISAKMP
          SPI Size              : 0
          Transform 1/4
            Transform ID        : IKE
            Encryption algorithm : Rijndael-cbc (aes)
            Key length           : 128
            Hash algorithm       : MD5
            Authentication method : Pre-Shared Key
            Group description     : MODP 1024
            Life type            : Seconds
            Life duration        : 43200
```

```

        Life type                : Kilobytes
        Life duration            : 50000
    Transform 2/4
        Transform ID             : IKE
        Encryption algorithm     : Rijndael-cbc (aes)
        Key length               : 128
        Hash algorithm           : SHA
        Authentication method    : Pre-Shared Key
        Group description        : MODP 1024
        Life type                : Seconds
        Life duration            : 43200
        Life type                : Kilobytes
        Life duration            : 50000
    Transform 3/4
        Transform ID             : IKE
        Encryption algorithm     : 3DES-cbc
        Hash algorithm           : MD5
        Authentication method    : Pre-Shared Key
        Group description        : MODP 1024
        Life type                : Seconds
        Life duration            : 43200
        Life type                : Kilobytes
        Life duration            : 50000
    Transform 4/4
        Transform ID             : IKE
        Encryption algorithm     : 3DES-cbc
        Hash algorithm           : SHA
        Authentication method    : Pre-Shared Key
        Group description        : MODP 1024
        Life type                : Seconds
        Life duration            : 43200
        Life type                : Kilobytes
        Life duration            : 50000
    VID (Vendor ID)
        Payload data length : 16 bytes
        Vendor ID   : 8f 9c c9 4e 01 24 8e cd f1 47 59 4c 28 4b 21 3b
        Description : SSH Communications Security QuickSec 2.1.0
    VID (Vendor ID)
        Payload data length : 16 bytes
        Vendor ID   : 27 ba b5 dc 01 ea 07 60 ea 4e 31 90 ac 27 c0 d0
        Description : draft-stenberg-ipsec-nat-traversal-01
    VID (Vendor ID)
        Payload data length : 16 bytes
        Vendor ID   : 61 05 c4 22 e7 68 47 e4 3f 96 84 80 12 92 ae cd
        Description : draft-stenberg-ipsec-nat-traversal-02
    VID (Vendor ID)
        Payload data length : 16 bytes
        Vendor ID   : 44 85 15 2d 18 b6 bb cd 0b e8 a8 46 95 79 dd cc
        Description : draft-ietf-ipsec-nat-t-ike-00
    VID (Vendor ID)
        Payload data length : 16 bytes
        Vendor ID   : cd 60 46 43 35 df 21 f8 7c fd b2 fc 68 b6 a4 48
        Description : draft-ietf-ipsec-nat-t-ike-02
    VID (Vendor ID)
        Payload data length : 16 bytes
        Vendor ID   : 90 cb 80 91 3e bb 69 6e 08 63 81 b5 ec 42 7b 1f
        Description : draft-ietf-ipsec-nat-t-ike-02
    VID (Vendor ID)
        Payload data length : 16 bytes
        Vendor ID   : 7d 94 19 a6 53 10 ca 6f 2c 17 9d 92 15 52 9d 56
        Description : draft-ietf-ipsec-nat-t-ike-03

```

### Explanation of Values

**Exchange type** : Main mode or aggressive mode

**Cookies** : A random number to identify the negotiation

**Encryption algorithm** : Cipher  
**Key length** : Cipher key length  
**Hash algorithm** : Hash  
**Authentication method** : Pre-shared key or certificate  
**Group description** : Diffie Hellman (DH) group  
**Life type** : Seconds or kilobytes  
**Life duration** : No of seconds or kilobytes  
**VID** : The IPsec software vendor plus what standards are supported eg.NAT-T

## Step 2. Server Responds to Client

A typical response from the server is shown below . This must contain a proposal that is identical to one of the choices from the client list above. If no match was found by the server then a "No proposal chosen" message will be seen, tunnel setup will fail and the *ikesnoop* command output will stop at this point.

```

IkeSnoop: Sending IKE packet to 192.168.0.10:500 Exchange type :
          Identity Protection (main mode) ISAKMP Version : 1.0

Flags      :
Cookies    : 0x6098238b67d97ea6 -> 0x5e347cb76e95a
Message ID : 0x00000000
Packet length : 224 bytes
# payloads  : 8
Payloads:
  SA (Security Association)
    Payload data length : 52 bytes
    DOI : 1 (IPsec DOI)
      Proposal 1/1
        Protocol 1/1
          Protocol ID      : ISAKMP
          SPI Size         : 0
          Transform 1/1
            Transform ID   : IKE
            Encryption algorithm : Rijndael-cbc (aes)
            Key length     : 128
            Hash algorithm  : MD5
            Authentication method : Pre-Shared Key
            Group description : MODP 1024
            Life type      : Seconds
            Life duration  : 43200
  VID (Vendor ID)
    Payload data length : 16 bytes
    Vendor ID   : 8f 9c c9 4e 01 24 8e cd f1 47 59 4c 28 4b 21 3b
    Description : SSH Communications Security QuickSec 2.1.0
  VID (Vendor ID)
    Payload data length : 16 bytes
    Vendor ID   : 27 ba b5 dc 01 ea 07 60 ea 4e 31 90 ac 27 c0 d0
    Description : draft-stenberg-ipsec-nat-traversal-01
  VID (Vendor ID)
    Payload data length : 16 bytes
    Vendor ID   : 61 05 c4 22 e7 68 47 e4 3f 96 84 80 12 92 ae cd
    Description : draft-stenberg-ipsec-nat-traversal-02
  VID (Vendor ID)
    Payload data length : 16 bytes
    Vendor ID   : 44 85 15 2d 18 b6 bb cd 0b e8 a8 46 95 79 dd cc
    Description : draft-ietf-ipsec-nat-t-ike-00
  VID (Vendor ID)
    Payload data length : 16 bytes
    Vendor ID   : cd 60 46 43 35 df 21 f8 7c fd b2 fc 68 b6 a4 48
    Description : draft-ietf-ipsec-nat-t-ike-02
  VID (Vendor ID)
    Payload data length : 16 bytes
    Vendor ID   : 90 cb 80 91 3e bb 69 6e 08 63 81 b5 ec 42 7b 1f
    Description : draft-ietf-ipsec-nat-t-ike-02
  
```

```

VID (Vendor ID)
  Payload data length : 16 bytes
  Vendor ID      : 7d 94 19 a6 53 10 ca 6f 2c 17 9d 92 15 52 9d 56
  Description   : draft-ietf-ipsec-nat-t-ike-03

```

### Step 3. Clients Begins Key Exchange

The server has accepted a proposal at this point and the client now begins a key exchange. In addition, NAT detection payloads are sent to detect if NAT is being used.

```

IkeSnoop: Received IKE packet from 192.168.0.10:500 Exchange type :
          Identity Protection (main mode) ISAKMP Version : 1.0
Flags      :
Cookies    : 0x6098238b67d97ea6 -> 0x5e347cb76e95a
Message ID : 0x00000000
Packet length : 220 bytes
# payloads  : 4
Payloads:
  KE (Key Exchange)
    Payload data length : 128 bytes
  NONCE (Nonce)
    Payload data length : 16 bytes
  NAT-D (NAT Detection)
    Payload data length : 16 bytes
  NAT-D (NAT Detection)
    Payload data length : 16 bytes

```

### Step 4. Server Sends Key Exchange Data

The Server now sends key exchange data back to the client.

```

IkeSnoop: Sending IKE packet to 192.168.0.10:500 Exchange type :
          Identity Protection (main mode) ISAKMP Version : 1.0
Flags      :
Cookies    : 0x6098238b67d97ea6 -> 0x5e347cb76e95a
Message ID : 0x00000000
Packet length : 220 bytes
# payloads  : 4
Payloads:
  KE (Key Exchange)
    Payload data length : 128 bytes
  NONCE (Nonce)
    Payload data length : 16 bytes
  NAT-D (NAT Detection)
    Payload data length : 16 bytes
  NAT-D (NAT Detection)
    Payload data length : 16 bytes

```

### Step 5. Client Sends Identification

The initiator sends the identification which is normally an IP address or the *Subject Alternative Name* if certificates are used.

```
IkeSnoop: Received IKE packet from 192.168.0.10:500 Exchange type :
          Identity Protection (main mode) ISAKMP Version : 1.0
Flags      : E (encryption)
Cookies    : 0x6098238b67d97ea6 -> 0x5e347cb76e95a
Message ID : 0x00000000
Packet length : 72 bytes
# payloads  : 3
Payloads:
  ID (Identification)
    Payload data length : 8 bytes
    ID : ipv4(any:0,[0..3]=192.168.0.10)
  HASH (Hash)
    Payload data length : 16 bytes
  N (Notification)
    Payload data length : 8 bytes
    Protocol ID : ISAKMP
    Notification : Initial contact
```

### Explanation of Above Values

**Flags** : E means encryption (it is the only flag used).

**ID** : Identification of the client

The *Notification* field is given as *Initial Contact* to indicate this is not a re-key.

### Step 6. Server ID Response

The server now responds with its own ID.

```
IkeSnoop: Sending IKE packet to 192.168.0.10:500 Exchange type :
          Identity Protection (main mode) ISAKMP Version : 1.0
Flags      : E (encryption)
Cookies    : 0x6098238b67d97ea6 -> 0x5e347cb76e95a
Message ID : 0x00000000
Packet length : 60 bytes
# payloads  : 2
Payloads:
  ID (Identification)
    Payload data length : 8 bytes
    ID : ipv4(any:0,[0..3]=192.168.10.20)
  HASH (Hash)
    Payload data length : 16 bytes
```

### Step 7. Client Sends IPsec Proposal List

Now the client sends the IPsec proposal list to the server. It will also contain the proposed host/networks allowed in the tunnel.

```
IkeSnoop: Received IKE packet from 192.168.0.10:500 Exchange type :
          Quick mode ISAKMP Version : 1.0
Flags      : E (encryption)
Cookies    : 0x6098238b67d97ea6 -> 0x5e347cb76e95a
Message ID : 0xaa71428f
Packet length : 264 bytes
# payloads  : 5
Payloads:
  HASH (Hash)
```

```

Payload data length : 16 bytes
SA (Security Association)
Payload data length : 164 bytes
DOI : 1 (IPsec DOI)
Proposal 1/1
  Protocol 1/1
    Protocol ID          : ESP
    SPI Size             : 4
    SPI Value            : 0x4c83cad2
  Transform 1/4
    Transform ID         : Rijndael (aes)
    Key length           : 128
    Authentication algorithm : HMAC-MD5
    SA life type         : Seconds
    SA life duration     : 21600
    SA life type         : Kilobytes
    SA life duration     : 50000
    Encapsulation mode   : Tunnel
  Transform 2/4
    Transform ID         : Rijndael (aes)
    Key length           : 128
    Authentication algorithm : HMAC-SHA-1
    SA life type         : Seconds
    SA life duration     : 21600
    SA life type         : Kilobytes
    SA life duration     : 50000
    Encapsulation mode   : Tunnel
  Transform 3/4
    Transform ID         : Blowfish
    Key length           : 128
    Authentication algorithm : HMAC-MD5
    SA life type         : Seconds
    SA life duration     : 21600
    SA life type         : Kilobytes
    SA life duration     : 50000
    Encapsulation mode   : Tunnel
  Transform 4/4
    Transform ID         : Blowfish
    Key length           : 128
    Authentication algorithm : HMAC-SHA-1
    SA life type         : Seconds
    SA life duration     : 21600
    SA life type         : Kilobytes
    SA life duration     : 50000
    Encapsulation mode   : Tunnel
NONCE (Nonce)
Payload data length : 16 bytes
ID (Identification)
Payload data length : 8 bytes
ID : ipv4(any:0,[0..3]=10.4.2.6)
ID (Identification)
Payload data length : 12 bytes
ID : ipv4_subnet(any:0,[0..7]=10.4.0.0/16)

```

### Explanation of Above Values

**Transform ID** : Cipher

**Key length** : Cipher key length

**Authentication algorithm** : HMAC ( Hash )

**Group description** : PFS and pfs group

**SA life type** : Seconds or Kilobytes

**SA life duration** : no seconds or kilobytes

**Encapsulation mode** : Could be transport, tunnel or UDP tunnel ( NAT-T )

**ID** : ipv4(any:0,[0..3]=10.4.2.6)



Here the first ID is the local network of the tunnel from the clients point of view and the second ID is the remote network. If it contains any netmask it is usually SA per net and otherwise it is SA per host.

### Step 8. Client Sends Proposal List

The server now responds with a matching IPsec proposal from the list sent by the client. As in step 2 above, if no match can be found by the server then a "No proposal chosen" message will be seen, tunnel setup will fail and the *ikesnoop* command output will stop here.

```
IkeSnoop: Sending IKE packet to 192.168.0.10:500 Exchange type :
          Quick mode ISAKMP Version : 1.0
Flags      : E (encryption)
Cookies    : 0x6098238b67d97ea6 -> 0x5e347cb76e95a
Message ID : 0xaa71428f
Packet length : 156 bytes
# payloads : 5
Payloads:
  HASH (Hash)
    Payload data length : 16 bytes
  SA (Security Association)
    Payload data length : 56 bytes
    DOI : 1 (IPsec DOI)
    Proposal 1/1
      Protocol 1/1
        Protocol ID      : ESP
        SPI Size         : 4
        SPI Value        : 0xafba2d15
      Transform 1/1
        Transform ID     : Rijndael (aes)
        Key length       : 128
        Authentication algorithm : HMAC-MD5
        SA life type     : Seconds
        SA life duration : 21600
        SA life type     : Kilobytes
        SA life duration : 50000
        Encapsulation mode : Tunnel
  NONCE (Nonce)
    Payload data length : 16 bytes
  ID (Identification)
    Payload data length : 8 bytes
    ID : ipv4(any:0,[0..3]=10.4.2.6)
  ID (Identification)
    Payload data length : 12 bytes
    ID : ipv4_subnet(any:0,[0..7]=10.4.0.0/16)
```

### Step 9. Client Confirms Tunnel Setup

This last message is a message from the client saying that the tunnel is up and running. All client/server exchanges have been successful.

```
IkeSnoop: Received IKE packet from 192.168.0.10:500 Exchange type :
          Quick mode ISAKMP Version : 1.0
Flags      : E (encryption)
Cookies    : 0x6098238b67d97ea6 -> 0x5e347cb76e95a
Message ID : 0xaa71428f
Packet length : 48 bytes
# payloads : 1
Payloads:
```

```
HASH (Hash)  
Payload data length : 16 bytes
```

## 9.4. PPTP/L2TP

The access by a client using a modem link over dial-up public switched networks, possibly with an unpredictable IP address, to protected networks via a VPN poses particular problems. Both the PPTP and L2TP protocols provide two different means of achieving VPN access from remote clients.

### 9.4.1. PPTP

Point to Point Tunneling Protocol (PPTP) is designed by the PPTP Forum, a consortium of companies that includes Microsoft. It is an OSI layer 2 "data-link" protocol (see Appendix F, *The OSI framework*) and is an extension of the older Point to Point Protocol (PPP), used for dial-up internet access. It was one of the first protocols designed to offer VPN access to remote servers via dial-up networks and is still widely used.

PPTP can be used in the VPN context to tunnel different protocols across the internet. Tunneling is achieved by encapsulating PPP packets in IP datagrams using Generic Routing Encapsulation (GRE). The client first establishes a connection to an ISP in the normal way using the PPP protocol and then establishes a TCP/IP connection across the internet to the Clavister Security Gateway which acts as the PPTP server. The ISP is not aware of the VPN since the tunnel extends from the PPTP server to the client. The PPTP standard does not define how data is encrypted. Encryption is usually achieved using the Microsoft Point-to-Point Encryption (MPPE) standard.

PPTP offers a convenient solution to client access that is simple to deploy. PPTP doesn't require the certificate infrastructure found in L2TP but instead relies on a username/password sequence to establish trust between client and server. The level of security offered by a non-certificate based solution is arguably one of PPTP's drawbacks. PPTP also presents some scalability issues with some PPTP servers restricting the number of simultaneous PPTP clients. Since PPTP doesn't use IPsec, PPTP connections can be NATed and NAT traversal is not required. PPTP has been bundled by Microsoft in its operating systems since Windows95 and therefore has a large number of clients with the software already installed.

#### Example 9.9. Setting up a PPTP server

This example shows how to setup a PPTP Network Server. The example assumes that you have already created certain address objects in the Address Book.

You will have to specify the IP address of the PPTP server interface, an outer IP address (that the PPTP server should listen to) and an IP pool that the PPTP server will use to give out IP addresses to the clients from.

##### *Clavister FineTune*

1. Go to **Interfaces > PPTP/L2TP Server**
2. Choose **New PPPT/L2TP Server** and the **PPPT/L2TP Properties** dialog will appear
3. Specify a name for the L2TP Server e.g. MyPPTPServer
4. Now enter:
  - **Inner IP Address:** ip\_lan
  - **Tunnel Protocol:** PPTP
  - **Outer Interface Filter:** any
  - **Outer Server IP:** ip\_wan
5. Under the **PPP Parameters** tab, select **pptp\_Pool** from the **IP Pool**
6. Under the **Add Route** tab, select **all\_nets** from **Allowed Networks**
7. Click **OK**

## 9.4.2. L2TP

Layer 2 Tunneling protocol (L2TP) is an IETF open standard that overcomes many of the problems of PPTP. Its design is a combination of Layer 2 Forwarding (L2F) protocol and PPTP, making use of the best features of both. Since the L2TP standard does not implement encryption, it is usually implemented with an IETF standard known as L2TP/IPsec, in which L2TP packets are encapsulated by IPsec. The client communicates with a Local Access Concentrator (LAC) and the LAC communicates across the internet with a L2TP Network Server (LNS). The Clavister Security Gateway acts as the LNS. The LAC is, in effect, tunneling data, such as a PPP session, using IPsec to the LNS across the internet. In most cases the client will itself act as the LAC.

L2TP is certificate based and therefore is simpler to administer with a large number of clients and arguably offers better security than PPTP. Unlike PPTP, it is possible to set up multiple virtual networks across a single tunnel. Being IPsec based, L2TP requires NAT traversal (NAT-T) to be implemented on the LNS side of the tunnel.

### Example 9.10. Setting up an L2TP server

This example shows how to setup a L2TP Network Server. The example presumes that you have created some address objects in the Address Book. You will have to specify the IP address of the L2TP server interface, an outer IP address (that the L2TP server should listen to) and an IP pool that the L2TP server will use to give out IP addresses to the clients from. The interface that the L2TP server will accept connections on is a virtual IPsec tunnel, not illustrated in this example.

#### *Clavister FineTune*

1. Go to **Interfaces > PPTP/L2TP Server**
2. Choose **New PPPT/L2TP Server** and the **PPPT/L2TP Server Properties** dialog will appear
3. Specify a name for the L2TP Server e.g. MyL2TPServer
4. Now enter:
  - **Inner IP Address:** ip\_l2tp
  - **Tunnel Protocol:** L2TP
  - **Outer Interface Filter:** l2tp\_ipsec
  - **Outer Server IP:** ip\_wan
5. Under the **PPP Parameters** tab, select **L2TP\_Pool** from the **IP Pool**
6. Under the **Add Route** tab, select **all\_nets** from **Allowed Networks**
7. Click **OK**

**Use User Authentication Rules** is enabled as default. To be able to authenticate the users using the PPTP tunnel you also need to configure authentication rules, which is not covered in this example.

### Example 9.11. Setting up an L2TP Tunnel

This example shows how to setup a fully working L2TP Tunnel and will cover many parts of basic VPN configuration. Before starting, you need to configure some address objects, e.g. the network that is going to be assigned to the L2TP clients. Proposal lists and PSK are needed as well. Here we will use the objects created in previous examples.

To be able to authenticate the users using the L2TP tunnel a local user database will be used.

1. Start with preparing a new Local User Database:

#### *Clavister FineTune*

1. Go to **Local Objects > User Databases**
2. Choose **New User Database** and the **User Database Properties** dialog will appear
3. Enter a name for the user database e.g. UserDB
4. Enter the new user database and choose **New User** from the context menu. The **User Properties** dialog will appear
5. Now enter:
  - **Username:** testuser
  - **Password:** mypassword
  - **Confirm Password:** mypassword
6. Click **OK**

Now we will setup the IPsec Tunnel, which will later be used in the L2TP section. As we are going to use L2TP, the Local Network is the same IP the L2TP tunnel will connect to, ip\_wan. Furthermore, the IPsec tunnel needs to be configured to dynamically add routes to the remote network when the tunnel is established.

2. Continue setting up the IPsec Tunnel:

#### **Clavister FineTune**

1. Go to **Interfaces > IPsec Tunnels**
2. Choose **New IPsec Tunnel** and the **IPsec Tunnel Properties** dialog will appear
3. Enter a name for the IPsec tunnel e.g. l2tp\_ipsec.
4. Now enter:
  - **Local Network:** ip\_wan
  - **Remote Network:** all-nets
  - **Remote Gateway:** none
  - **Encapsulation Mode:** Transport
  - **IKE Proposal List:** ike-roamingclients
  - **IPsec Proposal List:** esp-l2tptunnel
5. Under the **Authentication** tab, select **Pre-shared Key**
6. Select **MyPSK** as the **Pre-shared Key**
7. Under the **Routing** tab, check the following:
  - **Allow DHCP over IPsec from single-host clients**
  - **Dynamically add route to the remote network when a tunnel is established**
8. Click **OK**

Now it is time to setup the L2TP Server. The inner IP address should be a part of the network which the clients are assigned IP addresses from, in this ip\_lan. The outer interface filter is the interface that the L2TP server will accept connections on, this will be the earlier created l2tp\_ipsec. Also a ProxyARP needs to be configured for the IP's used by the L2TP Clients.

3. Setup the L2TP Tunnel:

#### **Clavister FineTune**

1. Go to **Interfaces > PPTP/L2TP Servers**
2. Choose **New PPTP/L2TP Server** and the **PPTP/L2TP Properties** dialog will appear
3. Enter a name for the L2TP tunnel e.g. l2tp\_tunnel.
4. Now enter:

- **Inner IP Address:** ip\_lan
  - **Tunnel Protocol:** L2TP
  - **Outer Interface Filter:** l2tp\_ipsec
  - **Outer Server IP:** ip\_wan
5. Under the **Authentication** tab, check **Use User Authentication Rules**
  6. Select **l2tp\_pool** from the **IP Pool**
  7. Under the **Add Route** tab, select **all-nets** for **Allowed Networks**
  8. For **ProxyARP** select the **lan** interface
  9. Click **OK**

In order to authenticate the users using the L2TP tunnel, a user authentication rule needs to be configured.

4. Next will be setting up the authentication rules:

#### ***Clavister FineTune***

1. Go to **User Authentication > Rules**
2. Choose **New Rule** from the context menu. The **Rule Properties** dialog will appear
3. Enter a name for the rule e.g. L2TP\_Auth
4. Now enter:
  - **Agent:** PPP
  - **Authentication Source:** Local
  - **Interface:** l2tp\_tunnel
  - **Outer Source IP:** all-nets
  - **Outer Destination IP:** ip\_wan
5. Click **OK**

When the other parts are done, all that is left is the rules. To let traffic trough from the tunnel, two certain IP rules should be added.

5. Finally, set up the rules:

#### ***Clavister FineTune***

1. Go to **Rules**
2. Choose **New Rule** and the **Rule Properties** dialog will appear
3. Enter a name for the rule e.g. AllowL2TP
4. Now enter:
  - **Action:** Allow
  - **Source Interface:** l2tp\_tunnel
  - **Source Network:** l2tp\_pool
  - **Destination Interface:** any
  - **Destination Network:** all-nets
5. Under the **Service** tab, select **All** from the pre-defined list
6. Click **OK**
7. Go to **Rules**
8. Choose **New Rule** from the context menu. The **Rule Properties** dialog will appear

9. Enter a name for the rule e.g. NATL2TP
10. Now enter:
  - **Action:** NAT
  - **Source Interface:** l2tp\_tunnel
  - **Source Network:** l2tp\_pool
  - **Destination Interface:** any
  - **Destination Network:** all-nets
11. Under the **Service** tab, select **All** from the pre-defined list
12. Click **OK**





---

# Chapter 10. Traffic Management

This chapter describes how CorePlus can manage network traffic.

- Traffic Shaping, page 220
- Traffic Rate Limiting with Threshold Rules, page 232
- Server Load Balancing, page 233

## 10.1. Traffic Shaping

### 10.1.1. Introduction

One of the major drawbacks of TCP/IP is the lack of true QoS functionality. Quality of Service in networks is the ability to guarantee and limit bandwidth for certain services and users.

Although there are protocols like DiffServ and other solutions that intend to offer QoS in large networks, none of the solutions have reached a high enough standard for large-scale usage.

Another fact is that most of the current Quality of Service solutions are application-based, that is, they work by having applications supplying the network with QoS-information. From a security standpoint, it is of course unacceptable that the applications (that is, the users) decide the priority of their own traffic within a network. In security-sensitive scenarios, where the users cannot be trusted, the network equipment should be the sole arbiter of priorities and bandwidth allocations.

The points listed above help explain why it is almost impossible to prioritize, guarantee to limit traffic in large and complex network topologies where different standards and different products exist. The Internet is a good example of such a network topology.

In well-delimited networks on the other hand, there are excellent possibilities to use different methods in order to control traffic. A well delimited network is defined mostly by the administrative limits, not the size of the network. The traffic in a MAN, and even in a very large WAN, could very well be managed, assuming that the network is designed in a homogeneous way.

CorePlus provides Quality of Service functionality by applying limits and guarantees to the network traffic itself, rather than trusting the applications and users to make these choices for themselves. It is hence well suited to manage bandwidth for a small LAN as well as in one or more choke points in large MANs or WANs.

### 10.1.2. Traffic Shaping Basics

The simplest way to obtain quality of service in a network, seen from a security as well as a functionality perspective, is to have the components in the network, not the applications, be responsible for network traffic control in well-defined choke points.

Traffic shaping works by measuring and queuing IP packets, in transit, with respect to a number of configurable parameters. Differentiated rate limits and traffic guarantees based on source, destination and protocol parameters can be created, much the same way gateway rules are implemented. Traffic shaping works by:

- Applying bandwidth limits by queuing packets that would exceed configured limits, and sending them later when the momentary demand for bandwidth is lower.
- Dropping packets if the packet buffers are full. The packet to be dropped should be chosen from those that are responsible for the "jam".

- Prioritizing traffic according to the administrator's choice; if the traffic in a higher priority increases while a communications line is full, traffic in lower priorities should be temporarily limited to make room for the high-priority traffic.
- Providing bandwidth guarantees. This is typically accomplished by treating a certain amount of traffic (the guaranteed amount) as a higher priority, and traffic exceeding the guarantee as the same priority as "any other traffic", which then gets to compete with the rest of the non-prioritized traffic.

Well-built traffic shapers do not normally work by queuing up immense amounts of data and then sorting out prioritized traffic to send before sending non-prioritized traffic. Rather, they attempt to measure the amount of prioritized traffic and then limit the non-prioritized traffic dynamically so that it won't interfere with the throughput of prioritized traffic.

## 10.1.3. Traffic Shaping in CorePlus

CorePlus offers extensive traffic shaping capabilities. Since any Clavister Security Gateway is a central and vital part of a network, there are many benefits of having it handle traffic control.

The Clavister traffic shaper has the following key features:

<b>Pipe based</b>	Traffic shaping in CorePlus is handled by a concept based on "pipes", where each pipe has several prioritizing, limiting and grouping possibilities. Individual pipes may be chained in different ways to construct bandwidth management units that far exceed the capabilities of one single pipe.
<b>Close integration with the gateway ruleset</b>	Each gateway rule may be assigned to one or more pipes, individually.
<b>Traffic prioritizing and bandwidth limiting</b>	Each pipe contains a number of priority levels, each with its own bandwidth limit specified in kilobits per second and/or packets per second. Limits may also be specified for the total of the pipe.
<b>Grouping</b>	<p>Traffic through a pipe can be automatically grouped into <i>pipe users</i>, where each pipe user can be configured in the same way as the main pipe.</p> <p>Traffic may be grouped with respect to a number of parameters, for instance source or destination IP network, IP address or port number.</p>
<b>Dynamic bandwidth balancing</b>	<p>The traffic shaper can be used to dynamically balance the bandwidth allocation of different pipe users if the pipe as a whole has exceeded its limits.</p> <p>This means that available bandwidth is evenly balanced with respect to the chosen grouping for the pipe.</p>
<b>Pipe chaining</b>	When pipes are assigned to rules, up to eight pipes may be connected to form a chain. This permits filtering and limiting to be handled in a very sophisticated manner.
<b>Traffic guarantees</b>	With the proper pipe configuration, traffic shaping may be used to guarantee bandwidth (and thereby quality) for traffic through the gateway.
<b>IPsec integration</b>	Bandwidth and priorities may be configured for IPsec VPN tunnels as well as for ordinary gateway rules.

## 10.1.4. Pipes Basics

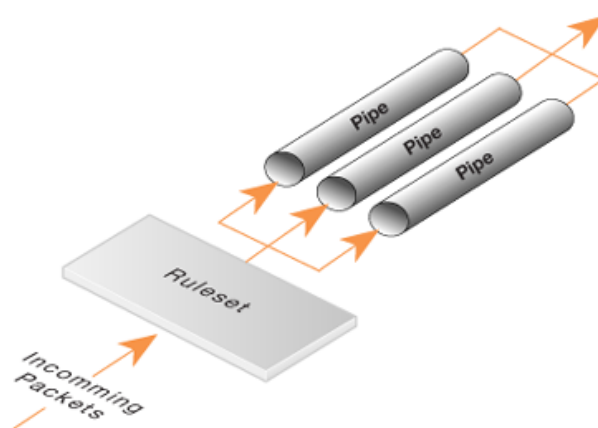
### 10.1.4.1. Definition of a Pipe

A Pipe is a central concept in the traffic shaping functionality of CorePlus and is the basis for all bandwidth control. Pipes are configured in the Pipes section of the gateway configuration.

Pipes are fairly simplistic, in that they do not know much about the types of traffic that pass through them, and they know nothing about direction. A pipe simply measures the traffic that passes through it and applies the configured limits in each precedence and/or user group.

Inbound network traffic is first filtered within the gateway IP rule-set, and is then passed to the pipe(s) specified in the matching rule. In the pipe, traffic is limited with respect to the configuration of the pipe and is then forwarded to its destination, or to the next pipe in a chain.

**Figure 10.1. Packet flow through pipes**



CorePlus is capable of handling hundreds of pipes simultaneously, but in reality, only a handful of pipes are required for most scenarios. The only time where you might end up with dozens and dozens of pipes are scenarios where you create individual pipes for each service (protocol, or client in ISP cases).

### 10.1.4.2. Simple Bandwidth Limit

The most basic use of pipes is simple bandwidth limits. This is also probably the only scenario that doesn't really require much planning.

In our first example, we will apply a bandwidth limit to only one direction, inbound traffic. This is the direction most likely to cause problems in an internet connection.

#### **Example 10.1. Applying a Simple Bandwidth Limit**

Begin with creating a simple pipe that limits all traffic that gets passed through it to 2 megabits per second, regardless of what traffic it is.

##### ***Clavister FineTune***

1. Select the **Traffic Management/Traffic Shaping** section in the tree view of the Security Editor.
2. Choose **New Pipe...** from the context menu. The **Pipe Properties** dialog box will be displayed.
3. Specify a suitable name for the object, for instance `std-in`.
4. Under **Pipe Limits**, enter 2000 in the **total** textbox (kilobits per sec.).

5. Click OK.

However, simply creating the pipe will not accomplish much; traffic actually needs to be passed *through* the pipe. This is done by assigning the pipe to an IP rule.

We will use the above pipe to limit inbound traffic. This limit will apply to the the actual data packets, and not the connections. In traffic shaping we're interested in the direction that data is being shuffled, not which computer initiated the connection.

Create a simple rule that allows everything from the inside, going out. We add the pipe that we created to the *return chain*. This means that the packets travelling in the *return direction* of this connection (outside-in) should pass through the "std-in" pipe.

#### **Clavister FineTune**

1. Select the **Traffic Management > Traffic Shaping > PipeRules** section in the tree view of the Security Editor.
2. Choose **New PipeRule...** from the context menu. The **PipeRule Properties** dialog box will be displayed.
3. Specify a suitable name for the piperule, for instance Outbound.
4. Now enter:
  - **Source Interface:** lan
  - **Source Network:** lannet
  - **Destination Interface:** wan
  - **Destination Network:** all-nets
5. Under the **Service** tab, select **all** in the Pre-defined dropdown list.
6. Under the **Traffic Shaping** tab, select the std-in pipe in **Return Chain** and click **Add**.
7. Click **OK**.

This setup limits all traffic from the outside (the Internet) to 2 megabits per second, much the same as if a 256 kbps Internet connection had been the bottleneck. No priorities are applied, nor any dynamic balancing.

### 10.1.4.3. Two-Way Bandwidth Limits

The previous example will only limit bandwidth in the inbound direction. We chose this direction simply because in most setups, it is the direction to first become full. Now, what if we want to limit bandwidth in both directions?

The answer is "simple! Apply the 2 Mbps limit in the forward direction as well!". Well, yes. But how?

Simply inserting "std-in" in the forward chain *will not work*. At least not the way you most likely want it to work. You probably want the 2 Mbps of outbound traffic to be separate from the 2 Mbps of inbound traffic, right?

So why doesn't the simple solution work? Well, as we've said before, pipes are simple things. If you try to pass 2 Mbps of outbound traffic through the pipe in addition to the 2 Mbps of inbound traffic, it would add up to 4 Mbps. Since the limit is at 2 Mbps, what you'd get is something like 1 Mbps in each direction.

However, you cannot just raise the total limit to 4 Mbps and hope for the best. Why? Again, pipes are simple things. This single pipe will not know that you mean 2 Mbps inbound and 2 Mbps outbound. You could just as well end up with 3 Mbps outbound and 1 Mbps inbound, since that, too, adds up to 4 Mbps.

Normally, the right way of controlling bandwidth in both directions is to use two pipes. One for inbound traffic and one for outbound traffic, each set to a 2 Mbps limit.

**Example 10.2. Applying a Two-Way Bandwidth Limit**

This example pre-assumes that you have gone through previous example.

Create a second pipe for outbound traffic:

**Clavister FineTune**

1. Select the **Traffic Management > Traffic Shaping > Pipe** section in the tree view of the Security Editor.
2. Choose **New Pipe...** from the context menu. The **Pipe Properties** dialog box will be displayed.
3. Specify a suitable name for the pipe, for instance std-out.
4. Under **Pipe Limits**, enter 2000 in the **total** textbox (kilobits per sec.).
5. Click OK.

When you've created your pipe for outbound bandwidth control, you simply add it to the forward pipe chain of the rule that you created in the previous example:

**Clavister FineTune**

1. Select the **Traffic Management > Traffic Shaping > PipeRules** section in the tree view of the Security Editor.
2. Right-click on the piperule you created in the previous example and choose **Properties**. The **PipeRule Properties** dialog box will be displayed.
3. Under the **Traffic Shaping** tab, select the std-out pipe in **Forward Chain** and click **Add**.
4. Click **OK**.

This results in all outbound connections being limited to 2 Mbps in each direction, closely emulating a normal 256 kbps Internet connection.

Of course, using the same pipe in both directions is perfectly legal, if what you want is "2 Mbps total, divided any way between forward and return data". Internet connections like these do exist, but normally you buy the same amount of bandwidth in both directions, where data flow in one direction doesn't affect the other direction.

### 10.1.4.4. Using Chains to create Differentiated Limits

Now, in the previous examples, all we've done is apply a simple static traffic limit for all outbound connections. What if we want to limit surfing further than the rest of the traffic?

Here, we could set up two "surf" pipes; inbound and outbound. However, the fact is, we most likely won't need to limit outbound traffic a whole lot, simply because surfing usually consists of short outbound requests followed by long inbound answers.

So, let's just set up a special "surf" pipe for inbound traffic and leave it at that:

Now that we have the pipe defined, what do we do with it? Well, first we will need to set up a rule that covers surfing and place it before the rule that covers "everything else". This way we can get surfing traffic to go through the specific pipes that we want it to, but still let everything else be handled by the "default" pipes we created earlier.

Copy the forward chain settings from the rule covering all "Standard" protocols.

Now, we'll have to figure out how to pass the return traffic through the "surf-in" pipe that we defined.

First, passing the surf traffic through the "surf-in" pipe in the return chain would seem a good enough idea, so let's start by doing that.

However, unfortunately, this will likely not get you the desired effect.

You will now have inbound traffic passing through two pipes: one that will forward 256 kbps, and one that will forward 128 kbps, for a total of 384 kbps of inbound traffic.

So, how do we limit the surfing traffic to 128 kbps without upsetting the grand total limits?

Simple: pass the inbound surf traffic through the std-in pipe as well.

Now, inbound surf traffic will first pass through the "surf-in" pipe, and get limited to 128 kbps. Then, it will get passed through the "std-in" pipe, along with the rest of the inbound traffic, which applies the 256 kbps total limit. So, if you're surfing for 128 kbps worth of bandwidth, those 128 kbps will occupy half of the std-in pipe, leaving only 128 kbps for the rest of the traffic, which is probably more along the lines of what you wanted.

If there is no surfing going on, all of the 256 kbps allowed through the std-in pipe will be available for all the rest of the traffic.



### Note

*This is not a traffic guarantee for web browsing. One could consider it a 128 kbps traffic guarantee for everything but web browsing, but for web browsing, the normal rules of first-come, first-served applies when competing for bandwidth. This may mean 128 kbps, but it may also mean the equivalent of a 2400 baud modem if your connection is sufficiently flooded.*

## 10.1.5. Priorities and Guarantees

### 10.1.5.1. Precedences

Each pipe contains eight *precedences*, or priority levels, numbered from 0 to 7. Each precedence may be seen as a separate queue where network traffic may be controlled. Precedence 0 is the least important precedence. Precedence 7 is the most important one.

**Figure 10.2. The Eight Pipe Precedences.**



### Note

*The respective precedences are not "special" in any way. Their meaning is only defined by the limits and guarantees that you configure. The difference is only in relative importance: traffic in precedence 2 will be passed on before traffic in precedence 0, traffic in precedence 4 before 2 and 0, and so on.*

In order to determine what precedence network traffic belongs to, each packet buffer is assigned a precedence number before it is sent into a pipe. The precedence assigned is controlled by the Rules section. This way, you can prioritize traffic by IP span, protocol number, port number, etc, the same way you normally filter traffic. This is described in greater detail later in this chapter.

When the pipe is configured, the number of precedences in the pipe can be defined by specifying a *Minimum precedence* and a *Maximum precedence*. The pipe will automatically adjust incoming packets to comply with these limits: a packet with a too low precedence is moved up to the minimum precedence. A packet with too high precedence is moved down to the maximum precedence. If a packet has no precedence, it is assigned the *Default precedence*.

The actual limiting of bandwidth is performed inside each precedence; separate bandwidth limits may be specified for each precedence. These limits may be specified in kilobits per second and/or packets per second.

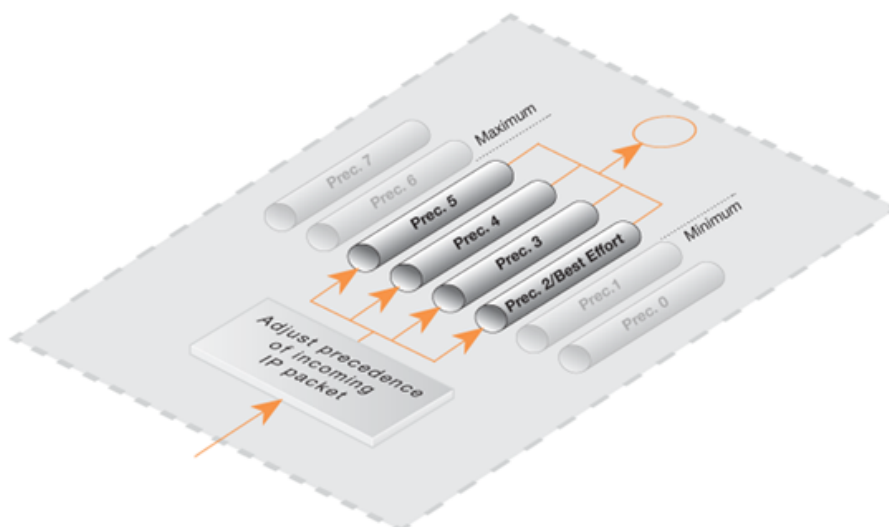
The precedence defined as the minimum precedence has a special functionality within the pipe: it acts as a best effort precedence.



### Note

*Traffic that exceeds the limit of a higher precedence will automatically be transferred into the best effort precedence, as long as there is room in the best effort precedence.*

**Figure 10.3. A Pipe defined with minimum precedence and maximum precedence.**



In addition to the limit per precedence, a limit for the pipe as a whole may also be specified, as you've seen in the previous example. When the total bandwidth through the pipe reaches the total limit, traffic will be prioritized depending on what precedence it belongs to. Higher precedences have a greater chance of making it through the pipe without queuing. However, if you are only using two precedences, choosing 4 and 6 rather than 0 and 2, or 0 and 6 if you like, will, of course, make no difference. The meaning of a precedence is only relative to traffic that passes in the other precedences, not to some external factor like, for instance, what is actually going on in the LAN outside the gateway, or on the other side of your Internet connection.

### 10.1.5.2. Applying Simple Priorities

Now, how can we use precedences to make some types of traffic more important than others? Let's continue work on our previous example, by giving SSH and Telnet traffic a higher priority than everything else passing through our pipes.

For this first example, we do not need to add or change anything in the Pipes section. First, we add a rule that covers SSH and Telnet traffic:

Copy the pipe settings from the "Standard" rule.

Previously, we had the precedence settings set at "Use defaults from first pipe". This means that the rule does not state that a specific precedence should be applied. Rather, it hands this decision over to the configuration of the first pipe in each respective pipe chain.

The default precedences of all pipes we have configured so far has been the lowest. To prioritize traffic above that, we instruct the rule to pass the packet to the pipe chains with a higher precedence. Let's choose 2.

Now, with this setup, SSH and Telnet traffic is simply prioritized before all other types of traffic. With these two low-throughput protocols, this behavior is likely not a problem.

However, if this had been real-time audio, it probably would have resulted in the audio streams using all available bandwidth and leaving none for surfing, DNS, FTP, and all the other protocols.

### 10.1.5.3. Simple Bandwidth Guarantees

Bandwidth guarantees aren't that much different from prioritizing certain types of traffic. All that really remains, is to limit the amount of high-priority bandwidth that may be used. The easiest but least flexible way of doing this is simply limiting how much gets to pass in the higher precedences of the default pipes.

To change the prioritized SSH and Telnet traffic from the previous example to a 96 kbps guarantee, you would simply change your *std-in* pipe to include a 96 kbps limit for precedence 2. Now, does this mean that your inbound SSH and telnet traffic is limited to 96 kbps? No, it does not.

As previously stated, excess traffic in precedences above the best-effort precedence gets passed to the best-effort precedence, which, in this example, is the lowest.

Again: Limits in precedences above the best-effort precedence will not actually limit the traffic. Such limits will only limit how much of the traffic gets to pass in that specific precedence.

So, in this case, the 96 kbps limit in precedence 2 means that you can pass up to 96 kbps worth of precedence 2 traffic to the *std-in* pipe, and this traffic will get through, unless there's traffic in even higher precedences.

If you attempt to pass more than 96 kbps of precedence 2 traffic, the excess traffic will have to compete with all the rest for the remaining bandwidth, and this competition is simple first-come, first-served, like any Internet connection. Grouping and balancing can improve this situation; more about that later on.

### 10.1.5.4. Differentiated Bandwidth Guarantees

As mentioned earlier, there is a slight problem with the previous way of implementing bandwidth guarantees: they are not very flexible.

What if, for instance, you want to give a specific 32 kbps guarantee to telnet traffic, and a specific 64 kbps guarantee to SSH traffic? You could set a 32 kbps limit for precedence 2, a 64 kbps limit for precedence 4, and pass the different types of traffic through each respective precedence. However, there are two obvious problems with this approach:

- Which traffic is more important? This question does not pose much of a problem here, but it becomes more pronounced as your traffic shaping scenario becomes more complex.
- The number of precedences is limited. This may not be sufficient in all cases, even barring the "which traffic is more important?" problem.

The solution here is to create two new pipes: one for telnet traffic, and one for SSH traffic, much like the "surf" pipe that we created earlier on.

First, remove the 96 kbps limit from the *std-in* pipe, then create two new pipes: "ssh-in" and "telnet-in". Set the default precedence for both pipes to 2, and the precedence 2 limits to 32 and 64 kbps, re-



spectively.

Then, split previously defined rule covering ports 22 through 23 into two rules, covering 22 and 23, respectively:

Keep the forward chain of both rules as "std-out" only. Again, to simplify this example, we concentrate only on inbound traffic, which is the direction that is the most likely to be the first one to fill up in client-oriented setups.

Set the return chain of the port 22 rule to "ssh-in" followed by "std-in". Set the return chain of the port 23 rule to "telnet-in" followed by "std-in". Set the priority assignment for both rules to "Use defaults from first pipe"; the default precedence of both the ssh-in and telnet-in pipes is 2. Using this approach rather than hard-coding precedence 2 in the ruleset, you can easily change the precedence of all SSH and telnet traffic by merely changing the default precedence of the "ssh-in" and "telnet-in" pipes.

You will notice that we did not set a total limit for the ssh-in and telnet-in pipes. We do not need to, since the total limit will be enforced by the "std-in" pipe at the end of the respective chains.

The ssh-in and telnet-in pipes act as a "priority filter": they make sure that no more than the reserved amount, 64 and 32 kbps, respectively, of precedence 2 traffic will reach std-in. SSH and telnet traffic exceeding their guarantees will reach std-in as precedence 0, the best-effort precedence of the std-in and ssh-in pipes.



### Note

*Here, the ordering of the pipes in the return chain is important. Should std-in appear before ssh-in and telnet-in, theN traffic will reach std-in as the lowest precedence only and hence compete for the 256 kbps of available bandwidth like any other traffic.*

## 10.1.5.5. Problems in Priorities and Guarantees

Guarantees are not just "guarantees". Guarantees and prioritized traffic work by limiting everything that is not prioritized. How are these limits calculated? In each pipe, the limit of lower priorities is calculated from the total limit, minus the current throughput of higher precedences.

### Set a total limit!

Bandwidth in lower precedences will not be throttled until a pipe thinks that it is full, i.e. passing as much traffic as the total limit states. After all, why should it throttle any sooner? If you have a 512 kbps pipe, passing 400 kbps of low priority traffic and 100 kbps of high priority traffic, there is no reason to throttle anything, since there is 12 kbps of bandwidth left.

So, in order to know how much to limit lower precedences, the pipe needs to know when it is full; it needs to know its total limit.

### Your total limits cannot be higher than your available connection bandwidth

If your pipe limits are set higher than the actual available bandwidth, the pipe will never know that the connection is full. If you have a 512 kbps connection, but your total pipe limits are set to 600 kbps, the pipe will think that it is not full. Therefore, it will not throttle the lower precedences.

### Measuring and shaping at the entrance of a choke point

If you are protecting the "entrance" to a network bottleneck, i.e. outbound data in your gateway, you can probably set the total limit very close to the bandwidth of your connection.

### Measuring and shaping at the exit of a choke point

If you're protecting the "exit" of a network bottleneck, i.e. inbound data in your gateway, you should probably set it a bit lower than the bandwidth of your connection. There are two risks involved in setting your limits so that they exactly match your inbound bandwidth:

- In the worst-case scenario, you could have a few stray packets per second consuming a fraction of your connection's bandwidth. As a result, your pipes will never think that they are full.
- There is also a much more real risk of having throttling adjustments taking too long a time, as the pipe will only see a "little" overload. If there is only a slight overload, then only slight adjustments will be made. This may result in very slow adaptations to new precedence distributions, possibly as slow as half a minute.
- Then, of course, there is the risk for connection overload. As you are shaping at the exit of the bottleneck, you have no control over what actually enters the bottleneck. As long as you are shaping well-behaved TCP, your traffic shaper will work, and even if internal clients stress the connection by sending phony ACKs, or whatever, they will not get much out of it, as the traffic shaper will just keep queuing packets destined for them.

However, shaping at the exit of a bottleneck does not protect against resource exhaustion attacks, such as DDoS or other floods. If someone is just bombarding you, they can overload your connection, and your traffic shaper cannot do anything about it. Sure, it will keep these extraneous packets from reaching the computers behind the shaper, but it will not protect your connection, and if your connection gets flooded, the attacker has won.

Some ISPs allow co-location, so if you believe that flooding is a realistic threat to you, you should consider co-locating your traffic shaper at the Internet side of your connection.

**You cannot guarantee bandwidth if you don't know what your available bandwidth is at all times**

For any traffic shaper to work, it needs to know the bandwidth passing through the choke point that it is trying to "protect".

If you are sharing your internet connection with other users or servers that are not under the control of your gateway, it is nearly impossible to guarantee, prioritize or balance bandwidth, simply because the gateway won't know how much bandwidth is available for your network. Simple limits will of course work, but guarantees, priorities and dynamic balancing will not.

**Watch for leaks!**

If you set out to protect and shape a network bottleneck, make sure that all traffic passing through that bottleneck passes through your pipes. If there's traffic going through your Internet connection that the pipes do not know about, they will not know when the Internet connection is full.

The problems resulting from leaks are exactly the same as in the cases described above. Traffic "leaking" through your gateway without being measured by your pipes will have the same effect as bandwidth consumed by parties outside of your control but sharing the same connection as you.

## 10.1.6. Grouping Users of a Pipe

### 10.1.6.1. Overview

If pipes were restricted to the functionality described so far, traffic would be limited without respect to source or destination. This mode of operation is likely sufficient for managing simple traffic limits and guarantees.

However the ability exists to group traffic within each pipe. This means that traffic will be classified and grouped with respect to the source or destination of each packet passing through the pipe.

Grouping may be performed on source network, source IP address, source port, destination network, destination IP address and destination port. In the network grouping cases, the network size may be specified. The port grouping cases also include the IP address, meaning that port 1024 of computer A is not the same "group" as port 1024 of computer B.

The benefit of using grouping is that additional bandwidth controls may be applied to each group. This means that if grouping is performed on, for example, IP address, the gateway can limit and guarantee bandwidth per IP address communicating through the pipe.

There are precedences in user groups, too. Bandwidth may be limited per precedence as well as for each group as a whole.

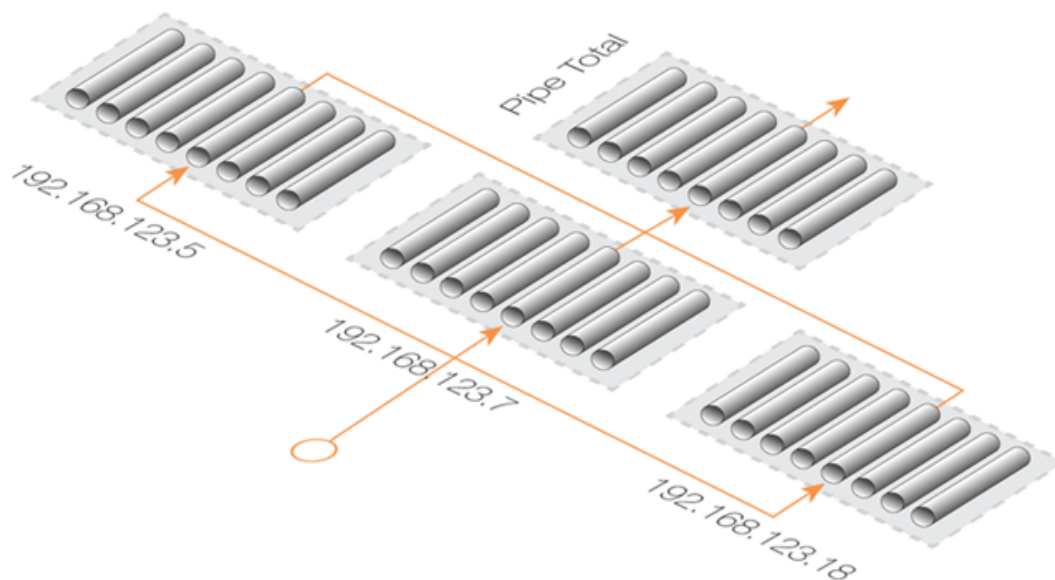


#### Note

*In many cases, we will refer to groups as "users", regardless of the group representing a physical person, a single connection, or an entire class C network.*

Bandwidth control first occurs per user and then continues with the pipe as a whole. A pipe with grouping enabled is illustrated in the drawing below.

**Figure 10.4. Example of a pipe with traffic in one precedence, grouped per IP address.**



### 10.1.6.2. Applying Per-User Limits and Guarantees

Once the users of a pipe are grouped, you can apply limits and guarantees to each user the same way you apply them to the pipe as a whole.

To expand on the previous example, we could, for instance, limit how much guaranteed bandwidth

each inside user gets for inbound SSH traffic. This keeps one single user from using up all available high-priority bandwidth.

First, we will have to figure out how to group the users of the ssh-in pipe. What we want to do is apply our limits to each user on the internal network. Considering that we are working with inbound packets, we will want to group per destination IP, so we change the grouping for the "ssh-in" pipe to "Per DestIP".

When the grouping is set, we can set per-user limits. In this case, we will set the precedence 1 limit to 16 kbps per user. This means that each user will get no more than a 16 kbps guarantee for their SSH traffic. If we wanted to, we could also limit the total bandwidth for each user to some value, maybe 40 kbps.

As you can see, we will run into problems if there are more than four users talking a lot of SSH simultaneously; 16 kbps times five is more than 64 kbps. The total limit for the pipe will still be in effect, and each user will have to compete for the available precedence 1 bandwidth the same way they have to compete for the lowest precedence bandwidth. Dynamic balancing can be used to improve this situation; more about that later.

For a better understanding of what is happening in a live setup, we recommend trying the "pipe -u <pipename>" console command. It will display a list of currently active users in each pipe.

### 10.1.6.3. Dynamic Bandwidth Balancing

As previously stated, per-user bandwidth may be limited by enabling grouping within a pipe. This may be used to ensure that one user cannot consume all of the available bandwidth.

But what if the bandwidth for the pipe as a whole has a limit, and that limit is exceeded?

In the previous example, the precedence 2 bandwidth limit per user is 16 kbps, and the precedence 2 limit for the pipe is 64 kbps. This means that up to four simultaneous users will get their fair share of high-precedence bandwidth.

If an additional user tries to talk SSH, the limit of 64 kbps will be exceeded. The results of this cannot be reliably predicted. Some users will still get their 16 kbps, some will not.

To prevent such situations, there is a feature called Dynamic Bandwidth Balancing. This algorithm ensures that the per-user bandwidth limits are dynamically lowered (and raised) in order to evenly balance the available bandwidth between the users of the pipe.

In the above sample, when the additional user begins to generate SSH traffic, the limit per user will be lowered to about 13 kbps (64 kbps divided by 5 users). Temporary restrictions such as these will be gradually removed, until the configured limit is reached, or until the pipe limits are exceeded, at which point the user limits will be lowered again. These dynamic adjustments take place 20 times per second, and will quickly adapt to changed bandwidth distributions.

Dynamic Bandwidth Balancing takes place within each precedence of a pipe individually. This means that if users are allotted a certain small amount of high priority traffic, and a larger chunk of best-effort traffic, all users will get their share of the high-precedence traffic as well as their fair share of the best-effort traffic.

## 10.2. Traffic Rate Limiting with Threshold Rules

### Overview

The Rate Limiting feature in CorePlus allows an administrator to put a limit on the number of new connections being opened across the Clavister Security Gateway per second. This is achieved by using a *Threshold Rule*. This is like a normal rule and becomes part of the gateway's ruleset.

The objective of Threshold Rules is to have a means of detecting abnormal connection activity as well as reacting to it. An example of a cause for such abnormal activity might be an internal host becoming infected with a virus that is making repeated connections to external IP addresses.

### Threshold Rule Actions

A Threshold Rule allows one of three actions when the specified rate limit is exceeded:

- **Ignore** - Do nothing
- **Audit** - Do nothing but log the event
- **Protect** - Blacklisting can be enabled

Logging would be the preferred option if the appropriate rate cannot be determined beforehand. Additionally the Threshold Rule can be applied to specific types of services eg. HTTP connections. If the option to drop traffic is chosen it is possible to elect that existing connections ie. those opened before the rate was exceeded, are not effected.

### 10.2.1. Rate limit blacklisting

Rate limiting can be configured so that the source that is causing the rate to be exceeded, is added automatically to a *Blacklist*.

The rate limiting rule can be network based which will cause the source network to be blacklisted instead of just an individual IP address. A host-based rule blacklists just the source IP. If the rate limiting rule is using a service then it is possible to block only that service.

This option is discussed further in Section 6.7, "Blacklisting Hosts and Networks".

## 10.3. Server Load Balancing

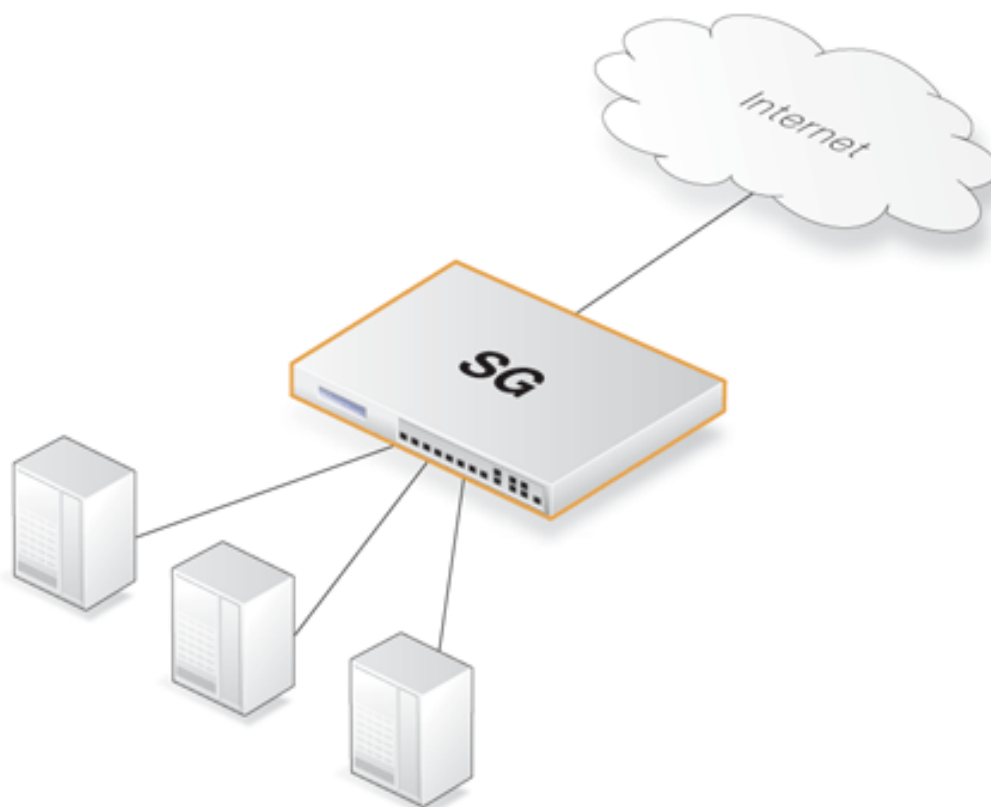
### 10.3.1. Introduction

The Clavister Server Load Balancing feature is a powerful tool that can improve several aspects of network applications:

- performance
- scalability
- reliability, and
- ease of administration.

Server Load Balancing (SLB) is common technique that allows network service demands to be shared among multiple servers. This improves the performance and scalability of any given application, by allowing a cluster of multiple servers (known as a "server farm") to handle far more requests than a single server.

**Figure 10.5. A Server Load Balancing configuration**



In addition to improving performance, SLB increases the reliability of network applications. By actively monitoring the servers sharing the load, SLB can detect when a server fails or becomes congested and will not direct any further requests to that server until it recovers.

SLB also means that network administrators can work on servers or applications without disrupting services. Individual servers can be restarted, upgraded, removed, or replaced, and new servers and

applications can be added or moved without affecting the rest of the server farm, or bringing down the application.

The combination of network monitoring and distributed load sharing also provides an extra level of protection against Denial Of Service (DoS) attacks.

The Clavister Server Load Balancing feature offers administrators the choice of several different algorithms to determine how to distribute the load. This allows them to tailor the load balancing rules to best suit the needs of their network.

Finally, the Server Load Balancing Module includes several distribution modes to provide support for a variety of different connection types. A typical SLB configuration can be represented as shown below:

## 10.3.2. Server Load Balancing Basics

There are four basic elements to the SLB technique. These are:

- Identify servers in a server farm
- Select the load distribution mode
- Select the SLB algorithm
- Select the monitoring method

### 10.3.2.1. Server Farms

A server farm is a cluster of servers set up to work as a single 'virtual server'. The servers that are to be treated as a single virtual server by the Load Balancing Module must be specified.

### 10.3.2.2. Load Distribution Modes

No single method of distributing server load is ideal for all services. Different types of services have different needs. In the SAT, you can configure rules for specific services. The SLB will then filter the packet flow according to these rules.

The Clavister Server Load Balancing Module supports the following distribution modes:

<b>Per-state Distribution</b>	In this mode, the SLB records the state of every connection. The entire session will then be distributed to the same server. This guarantees reliable data transmission for that session.
<b>IP Address Stickiness</b>	In this mode, all connections from a specific client will be sent to the same server. This is particularly important for SSL services such as https, which require a consistent connection to the same host.
<b>Network Stickiness</b>	This mode is similar to IP stickiness except that by using a subnet mask, you can specify a range of hosts in a subnet.

### 10.3.2.3. Distribution Algorithms

There are several ways to determine how a load is shared across a server farm. The Clavister Server Load Balancing feature supports the following algorithms:

<b>Round Robin</b>	The algorithm distributes new incoming connections to a list of servers on a rotating basis. For the first connection, the algorithm picks a server from
--------------------	--

the farm randomly, and assigns the connection to it. For subsequent connections, the algorithm cycles through the server list and redirects the load to servers in order. Regardless of each server's capability and other aspects, for instance, the number of existing connections on a server or its response time, all the available servers in the farm take turns to be assigned the next new connection.

This algorithm ensures that all servers receive an equal number of requests, therefore it is most suited to server farms where all servers have an equal capacity and the processing loads of all requests are likely to be similar.

#### Connection Rate

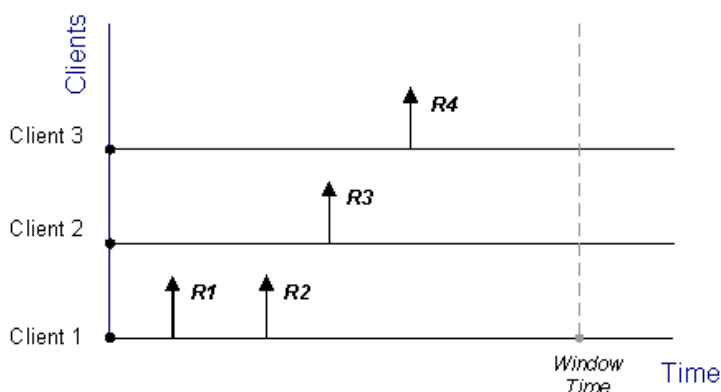
This algorithm considers the number of requests that each server has received over a certain timeframe. The SLB sends the next request to the server that has received the lowest number of connections in that time. The administrator is able to specify the timeframe to use with this algorithm.

If the Connection Rate algorithm is used without stickiness, it will behave as a Round Robin algorithm that allocates new connections to servers in an orderly fashion. It will also behave as the Round Robin algorithm if there always are clients with a new IP address that makes one connection. The real benefit is when using Connection Rate together with stickiness and clients make multiple connections. Then Connection Rate will take effect so that the distribution of new connections is as even as possible among the servers. Before the interval reaches the specified Idle Timeout of stickiness, new incoming connections from the same IP address as a previous connection are assigned to the same server. The connection with a new address will be redirected to a server with the lowest connection rate. The algorithm aims to minimize the new connection load for a server, but the distribution may get uneven if a client from a single IP is sending lots of new connections in a short time and the other servers does not get as many new connections.

In the management interface, the window time is the variable to use for counting the number of seconds back in time to summarize the number of new connections for connection-rate algorithm. As default value, 10 is used so that the number of new connections which were made to each server in the last 10 seconds will be memorized.

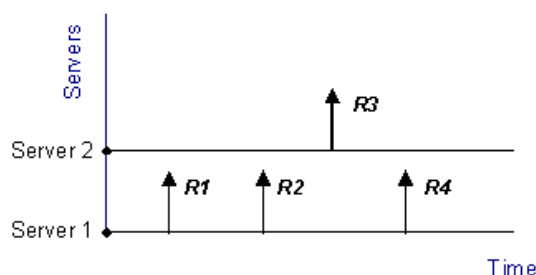
An example is shown in the figure below. In this example, the Clavister Security Gateway is responsible for balancing connections from 3 clients with different addresses to 2 servers. Stickiness is set.

**Figure 10.6. Connections from Three Clients**

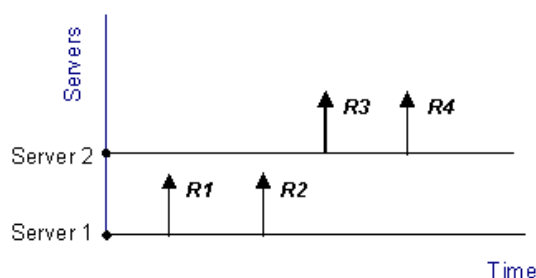


When the Round Robin algorithm is used, the first arriving requests R1 and R2 from Client 1 are both assigned to one server, say Server 1, according to stickiness. The next request R3 from Client 2 is then routed to Server 2. When R4 from Client 3 arrives, Server 1 gets back its turn again and will be assigned with R4.



**Figure 10.7. Stickiness and Round-Robin**

If Connection Rate is applied instead, R1 and R2 will be sent to the same server because of stickiness, but the subsequent requests R3 and R4 will be routed to another server since the number of new connections on each server within the Window Time span is counted in for the distribution.

**Figure 10.8. Stickiness and Connection Rate**

Regardless which algorithm is chosen, if a server goes down, traffic will be sent to other servers. And when the sever comes back online, it can automatically be placed back into the server farm and start getting requests again.

### 10.3.2.4. Server Health Monitoring

Server Health Monitoring is used to perform continuous checks on the condition of the individual servers in the server farm. The SLB can monitor at different OSI layers to check the connection rate of each server and its current state. Regardless of the algorithm in use, if a server fails, the SLB will not send it any more requests until it recovers.

The SLB will use the default routing table unless the administrator sets a specific routing table location.

Clavister Server Load Balancing provides the following monitoring modes:

#### ICMP Ping

This works at OSI layer 3. The SLB will ping the IP address of each individual server in the server farm. This will detect any failed servers.

#### TCP Connection

This works at OSI layer 4. The SLB attempts to connect to a specified port on each server. For example, if a server is specified as running web services on port 80, the SLB will send a TCP SYN request to that port. If the SLB does not receive a TCP SYN/ACK from the server, the SLB will mark port 80 on that server as down. The SLB can recognize no response, normal re-

sponse, and closed port response from servers.



---

# Chapter 11. High Availability

This chapter describes the high availability fault-tolerance feature in Clavister Security Gateways.

- Overview, page 239
- How rapid failover is accomplished, page 241
- Things to keep in mind, page 243

## 11.1. Overview

Clavister High Availability works by adding a back-up Clavister Security Gateway to an existing gateway. The back-up gateway has the same configuration as the primary gateway. It will stay inactive, monitoring the primary gateway, until it deems that the primary gateway is no longer functioning, at which point it will become active and assume the active role in the cluster. When the other gateway regains full functionality, the backup will assume a passive role, monitoring the now active gateway.

The hardware of the back-up gateway does not need to exactly match the hardware of the primary gateway. However, as role switches are not done unnecessarily, either gateway may stay active for an extended time, regardless of which one was originally the primary gateway. It is therefore recommended to use hardware of similar performance to avoid throughput degradation when a less-capable unit assumes the active role.

Throughout this chapter, the phrases "master gateway" and "primary gateway" are used interchangeably, as are the phrases "slave gateway" and "back-up gateway".

### What High Availability can do

Clavister High Availability will provide a redundant, state-synchronized gateway solution. This means that the state of the active gateway, i.e. connection table and other vital information, is continuously copied to the inactive gateway. When the cluster fails over to the inactive gateway, it knows which connections are active, and communication traffic can continue to flow uninterrupted.

The failover time is typically about one second; well within the scope for the normal TCP retransmit timeout, which is normally in excess of one minute. Clients connecting through the gateway will merely experience the failover as a slight burst of packet loss. TCP will, as it does in such situations, retransmit the lost packets within a second or two, and continue communication.

### What High Availability can *not* do

Adding redundancy to Clavister Security Gateway installations will eliminate one of the single points of failure in the communication path. However, it is not a panacea for all possible communication failures.

Typically, the gateway is far from the only single point of failure. Redundancy for routers, switches, and Internet connections are also issues that need to be examined.

Clavister High Availability clusters will not create a load-sharing cluster. One gateway will be active, and the other will be inactive. Multiple back-up gateways cannot be used in a cluster. Only two gateways, a "master" and a "slave", are supported. As is the case with all other gateways supporting stateful failover, Clavister High Availability will only work between two Clavister Security Gateways. As the internal workings of different manufacturer's gateways, and, indeed, different major versions of the same gateway, can be radically different, there is no way of communicating "state" to something which has a completely different comprehension of what "state" means.

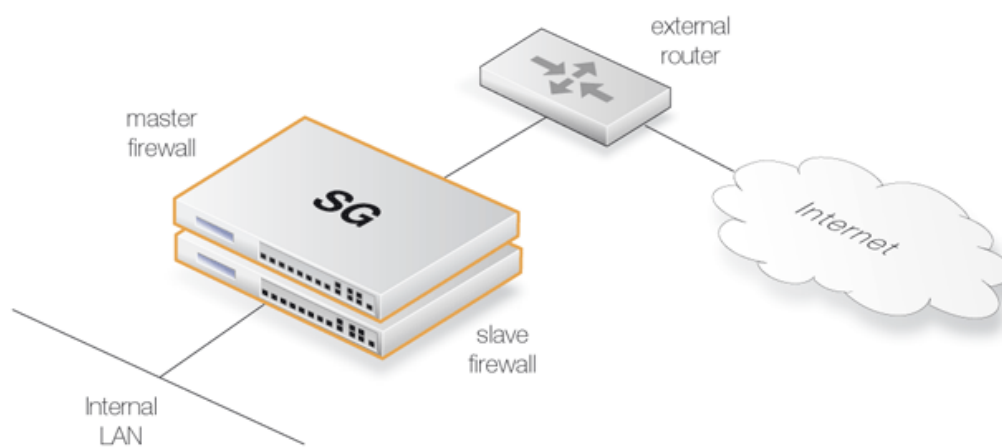
Broken interfaces will not be detected by the current implementation of Clavister High Availability,

unless they are broken to the point where the gateway cannot continue to run. This means that failover will not occur if the active gateway can communicate "being alive" to the inactive gateway through any of its interfaces, even though one or more interfaces may be inoperative.

## High Availability Setup Example

In a high availability setup, all the interfaces of the primary gateway need to be present on the backup gateway and be connected to the same networks. As previously mentioned, failover is not done unnecessarily, so either gateway may maintain the active role of the cluster for an extended period of time. Hence, connecting some equipment to only the "master" or only the "slave" gateway is bound to produce unwanted results. An example of a High Availability setup is shown below.

**Figure 11.1. High Availability Setup Example**



In this illustration, both gateways are connected to the internal as well as the external network. If there are more networks, for instance one or more demilitarized zones, or internal network segments, both gateways will also have to be connected to such networks; just connecting the "master" to a network will most likely lead to a loss of connectivity for extended periods of time.

## 11.2. How rapid failover is accomplished

This section will detail the outward-visible characteristics of the failover mechanism, and how the two gateways work together to create a high availability cluster with very low failover times.

For each cluster interface, there are three IP addresses:

- Two "real" IP addresses; one for each gateway. These addresses are used to communicate with the gateways themselves, i.e. for remote control and monitoring. They should not be associated in any way with traffic flowing through the cluster; if either gateway is inoperative, the associated IP address will simply be unreachable.
- One "virtual" IP address; shared between the gateways. This is the IP address to use when configuring default gateways and other routing related matters. It is also the address used by dynamic address translation, unless the configuration explicitly specifies another address.

There is not much to say about the real IP addresses; they will act just like gateway interfaces normally do. You can ping them or remote control the gateways through them if your configuration allows it. ARP queries for the respective addresses are answered by the gateway that owns the IP address, using the normal hardware address, just like normal IP units do.

### 11.2.1. Shared IP addresses and Failover

Both gateways in the cluster know about the shared IP address. ARP queries for the shared IP address, or any other IP address published via the ARP configuration section or through Proxy ARP, will be answered by the active gateway.

The hardware address of the shared IP address, and other published addresses for that matter, are not related to the hardware addresses of the gateway interfaces. Rather, it is constructed from the cluster ID, on the following form: 10-00-00-C1-4A-nn, where nn is the Cluster ID' configured in the Settings section.

As the shared IP address always has the same hardware address, there will be no latency time in updating ARP caches of units attached to the same LAN as the cluster when failover occurs.

When a gateway discovers that its peer is no longer operational, it will broadcast a number of ARP queries, using the shared hardware address as sender address, on all interfaces. This causes switches and bridges to re-learn where to send packets destined for the shared hardware address in a matter of milliseconds.

Hence, the only real delay in the failover mechanism is detecting that a gateway is no longer operational.

The activation messages (ARP queries) described above are also broadcast periodically to ensure that switches won't forget where to send packets destined for the shared hardware address.



#### **Note**

*The shared IP address should not be used for remote management or monitoring purposes. When using, for example, Netcon or SNMP for remote management of the Clavister Security Gateways in an HA configuration, the individual IP addresses of the gateways should be used.*

### 11.2.2. Cluster heartbeats

CorePlus detects that the peer system is no longer operational when it can no longer detect "cluster heartbeats" from its peer.

Currently, CorePlus will send five cluster heartbeats per second.

When three heartbeats are missed, i.e. after 0.6 seconds, the peer will be deemed inoperative.

So, why not make it even faster? Maybe send a hundred heartbeats per second and declare a gateway inoperative after missing only two of them? This would after all result in a 0.02-second failover time.

The problem with detection times less than one tenth of a second is that such delays may occur during normal operation. Just opening a file, on either gateway, could result in delays long enough to cause the inactive system to go active, even though the other is still active; a clearly undesirable situation.

Cluster heartbeats have the following characteristics:

- The source IP is the interface address of the sending gateway
- The destination IP is the shared IP address
- The IP TTL is always 255. If CorePlus receives a cluster heartbeat with any other TTL, it is assumed that the packet has traversed a router, and hence cannot be trusted at all.
- It is an UDP packet, sent from port 999, to port 999.
- The destination MAC address is the ethernet multicast address corresponding to the shared hardware address, i.e. 11-00-00-C1-4A-nn. Link-level multicasts were chosen over normal unicast packets for security reasons: using unicast packets would have meant that a local attacker could fool switches to route the heartbeats somewhere else, causing the peer system to never hear the heartbeats.

## 11.2.3. The synchronization interface

Both gateways are connected to each other by a separate synchronization connection; normal network cards are used, although they are dedicated solely to this purpose.

The active system continuously sends state update messages to its peer, informing it of connections that are opened, connections that are closed, state and life time changes in connections, etc.

When the active system ceases to function, for whatever reason and for even a short time, the cluster heartbeat mechanism described above will cause the inactive system to go active. Since it already knows about all open connections, communication can continue to flow uninterrupted.

## 11.3. Things to keep in mind

Even though a high availability cluster will behave like a single gateway in most respects, there are some things which should be kept in mind when managing and configuring it.

### 11.3.1. Communicating with two gateways

Clavister FineTune will automatically upload configurations to both gateways for you, but you will be communicating with two gateways.

#### Statistics are not shared

The Real-time Monitor will not automatically track the active gateway. If, all of a sudden, you're looking at a Real-time Monitor graph where nothing but the connection count is moving, your cluster has likely failed over to the other gateway.

SNMP statistics are also not shared. SNMP managers have no failover capabilities. Thus, you will need to poll both gateways in the cluster.

#### Logs come from two gateways

Log data will be coming from two gateways. This means that you will have to configure your log receiver to receive logs from both gateways. It also means that all your log queries will likely have to include both gateways as sources, which will give you all the log data in one result view. Normally, the inactive gateway won't be sending log entries about live traffic, so the output will likely look much the way it did with only one gateway.

### 11.3.2. Configuration issues

When configuring High Availability clusters, there are a number of things to keep in mind in order to avoid unnecessary pitfalls.

#### Changing the cluster ID

By changing the cluster ID, you actually doing two things:

- Changing the hardware address of the shared IPs. This will cause problems for all units attached to the local LAN, as they will keep the old hardware address in their ARP caches until it times out. Such units will have to have their ARP caches flushed.
- You will also break the connection between the gateways in the cluster for as long as they are using different configurations. This will cause both gateways to go active at the same time.

In short, changing the cluster ID unnecessarily is not a good idea.

After the configuration has been uploaded to both gateways, the ARP caches of vital units will have to be flushed in order to restore communication.

#### Never use the unique IPs for live traffic

The unique IP addresses of the gateways cannot safely be used for anything but managing the gateways.

Using them for anything else such as for source IPs in dynamically NATed connections or publishing services on them, will inevitably cause problems, as unique IPs will disappear when the gateway it belongs to does.





---

# Chapter 12. Advanced Settings

This chapter describes all CorePlus's configurable advanced settings. Settings are divided up into the following categories:

- IP Level Settings, page 246
- TCP Level Settings, page 249
- ICMP Level Settings, page 253
- ARP Settings, page 254
- Stateful Inspection Settings, page 257
- Connection Timeouts, page 259
- Size Limits by Protocol, page 260
- Fragmentation Settings, page 262
- Local Fragment Reassembly Settings, page 266
- VLAN Settings, page 267
- SNMP Settings, page 268
- DHCP Settings, page 269
- DHCPRelay Settings, page 270
- DHCPServer Settings, page 271
- IPsec Settings, page 272
- Transparent Mode Settings, page 273
- Logging Settings, page 275
- High Availability Settings, page 276
- Time Synchronization Settings, page 277
- DNS Client Settings, page 279
- Remote Administration Settings, page 280
- HTTP Poster Settings, page 281
- Hardware Performance Settings, page 282
- PPP Settings, page 283
- RADIUS Accounting Settings, page 284
- IDP, page 285
- Multicast Settings, page 286
- Hardware Monitor Settings, page 288
- Packet Re-assembly Settings, page 290
- Miscellaneous Settings, page 291

## 12.1. IP Level Settings

### LogChecksumErrors

Logs occurrences of IP packets containing erroneous checksums. Normally, this is the result of the packet being damaged during network transport. All network units, both routers and workstations, drop IP packets that contain checksum errors. However, it is highly unlikely for an attack to be based on illegal checksums.

Default: *Enabled*

### LogNonIP4

Logs occurrences of IP packets that are not version 4. CorePlus only accepts version 4 IP packets; everything else is discarded.

Default: *256*

### LogReceivedTTL0

Logs occurrences of IP packets received with the "Time To Live" (TTL) value set to zero. Under no circumstances should any network unit send packets with a TTL of 0.

Default: *Enabled*

### Block0000Src

Block 0.0.0.0 as source address.

Default: *Drop*

### Block0Net

Block 0.\* as source addresses.

Default: *DropLog*

### Block127Net

Block 127.\* as source addresses.

Default: *DropLog*

### BlockMulticastSrc

Block multicast both source addresses (224.0.0.0 - 255.255.255.255).

Default: *DropLog*

### TTLMin

The minimum TTL value accepted on receipt.

Default: *3*

### TTLOnLow

Determines the action taken on packets whose TTL falls below the stipulated TTLMin value.

Default: *DropLog*

## DefaultTTL

Indicates which TTL CorePlus is to use when originating a packet. These values are usually between 64 and 255.

Default: *255*

## LayerSizeConsistency

Verifies that the size information contained in each "layer" (Ethernet, IP, TCP, UDP, ICMP) is consistent with that of other layers.

Default: *ValidateLogBad*

## IPOptionSizes

Verifies the size of "IP options". These options are small blocks of information that may be added to the end of each IP header. This function checks the size of well-known option types and ensures that no option exceeds the size limit stipulated by the IP header itself.

Default: *ValidateLogBad*

## ILOPT\_SR

Indicates whether source routing options are to be permitted. These options allow the sender of the packet to control how the packet is to be routed through each router and gateway. These constitute an enormous security risk. CorePlus never obeys the source routes specified by these options, regardless of this setting.

Default: *DropLog*

## ILOPT\_TS

Time stamp options instruct each router and gateway on the packet's route to indicate at what time the packet was forwarded along the route. These options do not occur in normal traffic. Time stamps may also be used to "record" the route a packet has taken from sender to final destination. CorePlus never enters information into these options, regardless of this setting.

Default: *DropLog*

## ILOPT\_OTHER

All options other than those specified above.

Default: *DropLog*

## DirectedBroadcasts

Indicates whether CorePlus will forward packets which are directed to the broadcast address of its directly connected networks. It is possible to achieve this functionality by adding lines to the Rules section, but it is also included here for simplicity's sake. This form of validation is faster than entries in the Rules section since it is more specialized.

Default: *DropLog*

## IPRF

Indicates what CorePlus will do if there is data in the "reserved" fields of IP headers. In normal circumstances, these fields should read 0. Used by OS Fingerprinting.

Default: *DropLog*

## StripDFOnSmall

Strip the Don't Fragment flag for packets equal to or smaller than the size specified by this setting.

Default: *65535 bytes*

## 12.2. TCP Level Settings

### TCPOptionSizes

Verifies the size of TCP options. This function acts in the same way as IPOptionSizes described above.

Default: *ValidateLogBad*

### TCPMSSMin

Determines the minimum permissible size of the TCP MSS. Packets containing maximum segment sizes below this limit are handled according to the next setting.

Default: *100 bytes*

### TCPMSSOnLow

Determines the action taken on packets whose TCP MSS option falls below the stipulated TCPMSSMin value. Values that are too low could cause problems in poorly written TCP stacks.

Default: *DropLog*

### TCPMSSMax

Determines the maximum permissible TCP MSS size. Packets containing maximum segment sizes exceeding this limit are handled according to the next setting.

Default: *1460 bytes*

### TCPMSSVPNMax

As is the case with TCPMSSMax, this is the highest Maximum Segment Size allowed. However, this settings only controls MSS in VPN connections. This way, CorePlus can reduce the effective segment size used by TCP in all VPN connections. This reduces TCP fragmentation in the VPN connection even if hosts do not know how to perform MTU discovery.

Default: *1400 bytes*

### TCPMSSOnHigh

Determines the action taken on packets whose TCP MSS option exceeds the stipulated TCPMSSMax value. Values that are too high could cause problems in poorly written TCP stacks or give rise to large quantities of fragmented packets, which will adversely affect performance.

Default: *Adjust*

### TCPMSSAutoClamping

Automatically clamp TCP MSS according to MTU of involved intafaces, in addition to TCPMSSMax.

Default: *Enabled*

### TCPMSSLogLevel

Determines when to log regarding too high TCP MSS, if not logged by TCPMSSOnHigh.

Default: *7000 bytes*

## TCPZeroUnusedACK

Determines whether CorePlus should set the ACK sequence number field in TCP packets to zero if it is not used. Some operating systems reveal sequence number information this way, which can make it easier for intruders wanting to hijack established connections.

Default: *Enabled*

## TCPZeroUnusedURG

Strips the URG pointers from all packets.

Default: *Enabled*

## TCPOPT\_WSOPT

Determines how CorePlus will handle window-scaling options. These are used to increase the size of the windows used by TCP; i.e. the amount of information that can be sent before the sender expects ACK. They are also used by OS Fingerprinting. WSOPT is a common occurrence in modern networks.

Default: *ValidateLogBad*

## TCPOPT\_SACK

Determines how CorePlus will handle selective acknowledgement options. These options are used to ACK individual packets instead of entire series, which can increase the performance of connections experiencing extensive packet loss. They are also used by OS Fingerprinting. SACK is a common occurrence in modern networks.

Default: *ValidateLogBad*

## TCPOPT\_TSOPT

Determines how CorePlus will handle time stamp options. As stipulated by the PAWS (Protect Against Wrapped Sequence numbers) method, TSOPT is used to prevent the sequence numbers (a 32-bit figure) from "exceeding" their upper limit without the recipient being aware of it. This is not normally a problem. Using TSOPT, some TCP stacks optimize their connection by measuring the time it takes for a packet to travel to and from its destination. This information can then be used to generate resends faster than is usually the case. It is also used by OS Fingerprinting. TSOPT is a common occurrence in modern networks.

Default: *ValidateLogBad*

## TCPOPT\_ALTCHKREQ

Determines how CorePlus will handle alternate checksum request options. These options were initially intended to be used in negotiating for the use of better checksums in TCP. However, these are not understood by any today's standard systems. As CorePlus cannot understand checksum algorithms other than the standard algorithm, these options can never be accepted. The ALTCHKREQ option is normally never seen on modern networks.

Default: *StripLog*

## TCPOPT\_ALTCHKDATA

Determines how CorePlus will handle alternate checksum data options. These options are used to transport alternate checksums where permitted by ALTCHKREQ above. Normally never seen on modern networks.

Default: *StripLog*

## TCPOPT\_CC

Determines how CorePlus will handle connection count options.

Default: *StripLogBad*

## TCPOPT\_OTHER

Specifies how CorePlus will deal with TCP options not covered by the above settings. These options usually never appear on modern networks.

Default: *StripLog*

## TCPSynUrg

Specifies how CorePlus will deal with TCP packets with SYN (Synchronize) flags and URG (Urgent data) flags both turned on. The presence of a SYN flag indicates that a new connection is in the process of being opened, and an URG flag means that the packet contains data requiring urgent attention. These two flags should not be turned on in a single packet as they are used exclusively to crash computers with poorly implemented TCP stacks.

Default: *DropLog*

## TCPSynPsh

Specifies how CorePlus will deal with TCP packets with SYN and PSH (Push) flags both turned on. The PSH flag means that the recipient stack should immediately send the information in the packet to the destination application in the computer. These two flags should not be turned on at the same time as it could pose a crash risk for poorly implemented TCP stacks. However, many Macintosh computers do not implement TCP correctly, meaning that they always send out SYN packets with the PSH flag turned on. This is why CorePlus normally removes the PSH flag and allows the packet through despite the fact that the normal setting should be dropping such packets.

Default: *StripSilent*

## TCPFinUrg

Specifies how CorePlus will deal with TCP packets with both FIN (Finish, close connection) and URG flags turned on. This should normally never occur, as you do not usually attempt to close a connection at the same time as sending "important" data. This flag combination could be used to crash poorly implemented TCP stacks and is also used by OS Fingerprinting.

Default: *DropLog*

## TCPUrg

Specifies how CorePlus will deal with TCP packets with the URG flag turned on, regardless of any other flags. Many TCP stacks and applications deal with Urgent flags in the wrong way and can, in the worst case scenario, cease working. Note however that some programs, such as FTP and MS SQL Server, nearly always use the URG flag.

Default: *StripLog*



## TCPECN

Specifies how CorePlus will deal with TCP packets with either the Xmas or Ymas flag turned on. These flags are currently mostly used by OS Fingerprinting.

Note: an upcoming standard called Explicit Congestion Notification also makes use of these TCP flags, but as long as there are only a few operating systems supporting this standard, the flags should be stripped.

Default: *StripLog*

## TCPRF

Specifies how CorePlus will deal with information present in the "reserved field" in the TCP header, which should normally be 0. This field is not the same as the Xmas and Ymas flags. Used by OS Fingerprinting.

Default: *DropLog*

## TCPNULL

Specifies how CorePlus will deal with TCP packets that do not have any of the SYN, ACK, FIN or RST flags turned on. According to the TCP standard, such packets are illegal and are used by both OS Fingerprinting and stealth port scanners, as some gateways are unable to detect them.

Default: *DropLog*

## 12.3. ICMP Level Settings

### ICMPSendPerSecLimit

Specifies the maximum number of ICMP messages CorePlus may generate per second. This includes ping replies, destination unreachable messages and also TCP RST packets. In other words, this setting limits how many Rejects per second may be generated by the Reject rules in the Rules section.

Default: *20 per second*

### SilentlyDropStateICMPErrors

Specifies if CorePlus should silently drop ICMP errors pertaining to statefully tracked open connections. If these errors are not dropped by this setting, they are passed to the ruleset for evaluation just like any other packet.

Default: *Enabled*

## 12.4. ARP Settings

### ARPMatchEnetSender

Determines if CorePlus will require the sender address at Ethernet level to comply with the hardware address reported in the ARP data.

Default: *DropLog*

### ARPQueryNoSenderIP

What to do with ARP queries that have a sender IP of 0.0.0.0. Such sender IPs are never valid in responses, but network units that have not yet learned of their IP address sometimes ask ARP questions with an "unspecified" sender IP.

Default: *DropLog*

### ARPSenderIP

Determines if the IP sender address must comply with the rules in the Access section.

Default: *Validate*

### UnsolicitedARPReplies

Determines how CorePlus will handle ARP replies that it has not asked for. According to the ARP specification, the recipient should accept these. However, because this can facilitate hijacking of local connections, it is not normally allowed.

Default: *DropLog*

### ARPRequests

Determines if CorePlus will automatically add the data in ARP requests to its ARP table. The ARP specification states that this should be done, but as this procedure can facilitate hijacking of local connections, it is not normally allowed. Even if ARPRequests is set to "Drop", meaning that the packet is discarded without being stored, CorePlus will, provided that other rules approve the request, reply to it.

Default: *Drop*

### ARPChanges

Determines how CorePlus will deal with situations where a received ARP reply or ARP request would alter an existing item in the ARP table. Allowing this to take place may facilitate hijacking of local connections. However, not allowing this may cause problems if, for example, a network adapter is replaced, as CorePlus will not accept the new address until the previous ARP table entry has timed out.

Default: *AcceptLog*

### StaticARPChanges

Determines how CorePlus will handle situations where a received ARP reply or ARP request would alter a static item in the ARP table. Of course, this is never allowed to happen. However, this setting does allow you to specify whether or not such situations are to be logged.

Default: *DropLog*

## ARPExpire

Specifies how long a normal dynamic item in the ARP table is to be retained before it is removed from the table.

Default: *900 seconds (15 minutes)*

## ARPExpireUnknown

Specifies how long CorePlus is to remember addresses that cannot be reached. This is done to ensure that CorePlus does not continuously request such addresses.

Default: *3 seconds*

## ARPMulticast

Determines how CorePlus is to deal with ARP requests and ARP replies that state that they are multicast addresses. Such claims are usually never correct, with the exception of certain load balancing and redundancy devices, which make use of hardware layer multicast addresses.

Default: *DropLog*

## ARPBroadcast

Determines how CorePlus is to deal with ARP requests and ARP replies that state that they are broadcast addresses. Such claims are usually never correct.

Default: *DropLog*

## ARPCacheSize

How many ARP entries there can be in the cache in total.

Default: *4096*

## ARPHashSize

So-called "hash tables" are used to rapidly look up entries in a table. For maximum efficiency, a hash should be twice as large as the table it is indexing, so if the largest directly-connected LAN contains 500 IP addresses, the size of the ARP entry hash should be at least 1000 entries.

Default: *512*

## ARPHashSizeVLAN

So-called "hash tables" are used to rapidly look up entries in a table. For maximum efficiency, a hash should be twice as large as the table it is indexing, so if the largest directly-connected LAN contains 500 IP addresses, the size of the ARP entry hash should be at least 1000 entries.

Default: *64*

## ARPIPCollision

Determines the behavior when receiving an ARP request with a sender IP address that collides with one already used on the receive interface. Possible actions: Drop or Notify.

Default: *Drop*

## 12.5. Stateful Inspection Settings

### ConnReplace

Allows new additions to CorePlus's connection list to replace the oldest connections if there is no available space.

Default: *ReplaceLog*

### LogOpenFails

In some instances where the Rules section determines that a packet should be allowed through, the stateful inspection mechanism may subsequently decide that the packet cannot open a new connection. One example of this is a TCP packet that, although allowed by the Rules section and not being part of an established connection, has its SYN flag off. Such packets can never open new connections. In addition, new connections can never be opened by ICMP messages other than ICMP ECHO (Ping). This setting determines if CorePlus is to log the occurrence of such packets.

Default: *Enabled*

### LogReverseOpens

Determines if CorePlus logs packets that attempt to open a new connection back through one that is already open. This only applies to TCP packets with the SYN flag turned on and to ICMP ECHO packets. In the case of other protocols such as UDP, there is no way of determining whether the remote peer is attempting to open a new connection.

Default: *Enabled*

### LogStateViolations

Determines if CorePlus logs packets that violate the expected state switching diagram of a connection, for instance, getting TCP FIN packets in response to TCP SYN packets.

Default: *Enabled*

### MaxConnections

Specifies how many connections CorePlus may keep open at any one time. Each connection consumes approximately 150 bytes RAM. When this settings is dynamic, CorePlus will try to use as many connections as is allowed by product.

Default: *<dynamic>*

### LogConnections

Specifies how CorePlus, will log connections:

- *NoLog* – Does not log any connections; consequently, it will not matter if logging is enabled for either Allow or NAT rules in the Rules section; they will not be logged. However, FwdFast, Drop and Reject rules will be logged as stipulated by the settings in the Rules section.
- *Log* – Logs connections in short form; gives a short description of the connection, which rule allowed it to be made and any NAT rules that apply. Connections will also be logged when they are closed.
- *LogOC* – As for Log, but includes the two packets that cause the connection to be opened and

closed. If a connection is closed as the result of a timeout, no ending packet will be logged

- *LogOCall* – Logs all packets involved in opening and closing the connection. In the case of TCP, this covers all packets with SYN, FIN or RST flags turned on
- *LogAll* – Logs all packets in the connection.

Default: *Log*

## 12.6. Connection Timeouts

The settings in this section specify how long a connection can remain idle, i.e. no data being sent through it, before it is automatically closed. Please note that each connection has two timeout values: one for each direction. A connection is closed if either of the two values reaches 0.

### ConnLife\_TCP\_SYN

Specifies how long a not yet been fully established TCP connection may idle before being closed.

Default: *60 seconds*

### ConnLife\_TCP

Specifies how long a fully established TCP connection may idle before being closed. Connections become fully established once packets with their SYN flags off have traveled in both directions.

Default: *262144 seconds*

### ConnLife\_TCP\_FIN

Specifies how long a TCP connection about to close may idle before finally being closed. Connections reach this state when a packet with its FIN flag on has passed in any direction.

Default: *80 seconds*

### ConnLife\_UDP

Specifies how long UDP connections may idle before being closed. This timeout value is usually low, as UDP has no way of signaling when the connection is about to close.

Default: *130 seconds*

### ConnLife\_Ping

Specifies how long a Ping (ICMP ECHO) connection can remain idle before it is closed.

Default: *8 seconds*

### ConnLife\_Other

Specifies how long connections using an unknown protocol can remain idle before it is closed.

Default: *130 seconds*

### ConnLife\_IGMP

Connection lifetime for IGMP

Default: *12 seconds*



## 12.7. Size Limits by Protocol

This section contains information about the size limits imposed on the protocols directly under IP level, i.e. TCP, UDP, ICMP, etc

The values specified here concern the IP data contained in packets. In the case of Ethernet, a single packet can contain up to 1480 bytes of IP data without fragmentation. In addition to that, there is a further 20 bytes of IP header and 14 bytes of Ethernet header, corresponding to the maximum media transmission unit on Ethernet networks of 1514 bytes.

### MaxTCPLen

Specifies the maximum size of a TCP packet including the header. This value usually correlates with the amount of IP data that can be accommodated in an unfragmented packet, since TCP usually adapts the segments it sends to fit the maximum packet size. However, this value may need to be increased by 20-50 bytes on some less common VPN systems.

Default: *1480*

### MaxUDPLen

Specifies the maximum size of a UDP packet including the header. This value may well need to be quite high, since many real-time applications use large, fragmented UDP packets. If no such protocols are used, the size limit imposed on UDP packets can probably be lowered to 1480 bytes.

Default: *60000 bytes*

### MaxICMPLen

Specifies the maximum size of an ICMP packet. ICMP error messages should never exceed 600 bytes, although Ping packets can be larger if so requested. This value may be lowered to 1000 bytes if you do not wish to use large Ping packets.

Default: *10000 bytes*

### MaxGRELen

Specifies the maximum size of a GRE packet. GRE, Generic Routing Encapsulation, has various uses, including the transportation of PPTP, Point to Point Tunneling Protocol, data. This value should be set at the size of the largest packet allowed to pass through the VPN connections, regardless of its original protocol, plus approx. 50 bytes.

Default: *2000 bytes*

### MaxESPLen

Specifies the maximum size of an ESP packet. ESP, Encapsulation Security Payload, is used by IPsec where encryption is applied. This value should be set at the size of the largest packet allowed to pass through the VPN connections, regardless of its original protocol, plus approx. 50 bytes.

Default: *2000 bytes*

### MaxAHLen

Specifies the maximum size of an AH packet. AH, Authentication Header, is used by IPsec where only authentication is applied. This value should be set at the size of the largest packet allowed to pass through the VPN connections, regardless of its original protocol, plus approx. 50 bytes.

Default: *2000 bytes*

## MaxSKIPLen

Specifies the maximum size of a SKIP packet.

Default: *2000 bytes*

## MaxOSPFLen

Specifies the maximum size of an OSPF packet. OSPF is a routing protocol mainly used in larger LANs.

Default: *1480*

## MaxIPIPLen

Specifies the maximum size of an IP-in-IP packet. IP-in-IP is used by Checkpoint Firewall-1 VPN connections when IPsec is not used. This value should be set at the size of the largest packet allowed to pass through the VPN connections, regardless of its original protocol, plus approx. 50 bytes.

Default: *2000 bytes*

## MaxIPCompLen

Specifies the maximum size of an IPComp packet.

Default: *2000 bytes*

## MaxL2TPLen

Specifies the maximum size of a Layer 2 Tunneling Protocol packet.

Default: *2000 bytes*

## MaxOtherSubIPLen

Specifies the maximum size of packets belonging to protocols that are not specified above.

Default: *1480 bytes*

## LogOversizedPackets

Specifies if CorePlus will log oversized packets.

Default: *Enabled*

## 12.8. Fragmentation Settings

IP is able to transport up to 65536 bytes of data. However, most media, such as Ethernet, cannot carry such huge packets. To compensate, the IP stack fragments the data to be sent into separate packets, each one given their own IP header and information that will help the recipient reassemble the original packet correctly.

However, many IP stacks are unable to handle incorrectly fragmented packets, a fact that can be exploited by intruders to crash such systems. CorePlus provides protection against fragmentation attacks in a number of ways.

### PseudoReass\_MaxConcurrent

Maximum number of concurrent fragment reassemblies. To drop all fragmented packets, set PseudoReass\_MaxConcurrent to 0.

Default: *1024*

### IllegalFrag

Determines how CorePlus will handle incorrectly constructed fragments. The term "incorrectly constructed" refers to overlapping fragments, duplicate fragments with different data, incorrect fragment sizes, etc. Possible settings include:

- *Drop* – Discards the illegal fragment without logging it. Also remembers that the packet that is being reassembled is "suspect", which can be used for logging further down the track.
- *DropLog* – Discards and logs the illegal fragment. Also remembers that the packet that is being reassembled is "suspect", which can be used for logging further down the track.
- *DropPacket* – Discards the illegal fragment and all previously stored fragments. Will not allow further fragments of this packet to pass through during ReassIllegalLinger seconds.
- *DropLogPacket* – As DropPacket, but also logs the event.
- *DropLogAll* – As DropLogPacket, but also logs further fragments belonging to this packet that arrive during ReassIllegalLinger seconds.

The choice of whether to discard individual fragments or disallow the entire packet is governed by two factors:

- It is safer to discard the whole packet.
- If, as the result of receiving an illegal fragment, you choose to discard the whole packet, attackers will be able to disrupt communication by sending illegal fragments during a reassembly, and in this way block almost all communication.

Default: *DropLog* – discards individual fragments and remembers that the reassembly attempt is "suspect".

### DuplicateFragData

If the same fragment arrives more than once, this can mean either that it has been duplicated at some point on its journey to the recipient or that an attacker is trying to disrupt the reassembly of the packet. In order to determine which is more likely, CorePlus compares the data components of the fragment. The comparison can be made in 2 to 512 random locations in the fragment, four bytes of each location being sampled. If the comparison is made in a larger number of samples, it is more

likely to find mismatching duplicates. However, more comparisons result in higher CPU load.

Default: *Check8* – compare 8 random locations, a total of 32 bytes

## FragReassemblyFail

Reassemblies may fail due to one of the following causes:

- Some of the fragments did not arrive within the time stipulated by the ReassTimeout or ReassTimeLimit settings. This may mean that one or more fragments were lost on their way across the Internet, which is a quite common occurrence.
- CorePlus was forced to interrupt the reassembly procedure due to new fragmented packets arriving and the system temporarily running out of resources. In situations such as these, old reassembly attempts are either discarded or marked as "failed".
- An attacker has attempted to send an incorrectly fragmented packet.

Under normal circumstances, you would not want to log failures as they occur frequently. However, it may be useful to log failures involving "suspect" fragments. Such failures may arise if, for example, the IllegalFragments setting has been set to Drop rather than DropPacket.

The following settings are available for FragReassemblyFail:

- *NoLog* - No logging is done when a reassembly attempt fails.
- *LogSuspect* - Logs failed reassembly attempts only if "suspect" fragments have been involved.
- *LogSuspectSubseq* - As *LogSuspect*, but also logs subsequent fragments of the packet as and when they arrive
- *LogAll* - Logs all failed reassembly attempts.
- *LogAllSubseq* - As *LogAll*, but also logs subsequent fragments of the packet as and when they arrive.

Default: *LogSuspectSubseq*

## DroppedFragments

If a packet is denied entry to the system as the result of the settings in the Rules section, it may also be worth logging individual fragments of that packet. The DroppedFragments setting specifies how CorePlus will act. Possible settings for this rule are as follows:

- *NoLog* – No logging is carried out over and above that which is stipulated in the ruleset.
- *LogSuspect* - Logs individual dropped fragments of reassembly attempts affected by "suspect" fragments.
- *LogAll* - Always logs individual dropped fragments.

Default: *LogSuspect*

## DuplicateFragments

If the same fragment arrives more than once, this can mean either that it has been duplicated at some point on its journey to the recipient or that an attacker is trying to disrupt the reassembly of the

packet. DuplicateFragments determines whether such a fragment should be logged. Note that DuplicateFragData can also cause such fragments to be logged if the data contained in them does not match up. Possible settings are as follows:

- *NoLog* - No logging is carried out under normal circumstances.
- *LogSuspect* - Logs duplicated fragments if the reassembly procedure has been affected by "suspect" fragments.
- *LogAll* - Always logs duplicated fragments.

Default: *LogSuspect*

## FragmentedICMP

Other than ICMP ECHO (Ping), ICMP messages should not normally be fragmented as they contain so little data that fragmentation should never be necessary. FragmentedICMP determines the action taken when CorePlus receives fragmented ICMP messages that are not either ICMP ECHO or ECHOREPLY.

Default: *DropLog*

## MinimumFragLength

MinimumFragLength determines how small all fragments, with the exception of the final fragment, of a packet can be. Although the arrival of too many fragments that are too small may cause problems for IP stacks, it is often not possible to set this limit too high. It is rarely the case that senders create very small fragments. However, a sender may 1480 byte fragments and a router or VPN tunnel on the route to the recipient subsequently reduce the effective MTU to 1440 bytes. This would result in the creation of a number of 1440 byte fragments and an equal number of 40 byte fragments. Because of potential problems this can cause, the default settings in CorePlus has been designed to allow the smallest possible fragments, 8 bytes, to pass. For internal use, where all media sizes are known, this value can be raised to 200 bytes or more.

Default: *8 bytes*

## ReassTimeout

A reassembly attempt will be interrupted if no further fragments arrive within ReassTimeout seconds of receipt of the previous fragment.

Default: *65 seconds*

## ReassTimeLimit

A reassembly attempt will always be interrupted ReassTimeLimit seconds after the first received fragment arrived.

Default: *90 seconds*

## ReassDoneLinger

Once a packet has been reassembled, CorePlus is able to remember this for a short period of time in order to prevent further fragments, e.g. old duplicate fragments, of that packet from arriving.

Default: *20 seconds*

## ReassIllegalLinger

Once a whole packet has been marked as illegal, CorePlus is able to retain this in its memory in order to prevent further fragments of that packet from arriving.

Default: *60 seconds*

## 12.9. Local Fragment Reassembly Settings

### LocalReass\_MaxConcurrent

Maximum number of concurrent local reassemblies.

Default: *256*

### LocalReass\_MaxSize

Maximum size of a locally reassembled packet.

Default: *10000*

### LocalReass\_NumLarge

Number of large ( over 2K) local reassembly buffers (of the above size).

Default: *32*

## 12.10. VLAN Settings

### UnknownVLANTags

What to do with VLAN packets tagged with an unknown ID.

Default: *DropLog*



## 12.11. SNMP Settings

### SNMPReqLimit

Maximum number of SNMP requests that will be processed each second.

Default: *100*

### SNMPSysContact

The contact person for the managed node.

Default: *"N/A"*

### SNMPSysName

The name for the managed node.

Default: *"N/A"*

### SNMPSysLocation

The physical location of the node.

Default: *"N/A"*

### SNMPifDescr

What to display in the SNMP MIB-II ifDescr variables.

Default: *Name*

### SNMPifAlias

What to display in the SNMP ifMIB ifAlias variables.

Default: *Hardware*

## 12.12. DHCP Settings

### DHCP\_MinimumLeaseTime

Minimum lease time (seconds) accepted from the DHCP server.

Default: *60*

### DHCP\_ValidateBcast

Require that the assigned broadcast address is the highest address in the assigned network.

Default: *Enabled*

### DHCP\_AllowGlobalBcast

Allow DHCP server to assign 255.255.255.255 as broadcast. (Non-standard.)

Default: *Disabled*

### DHCP\_UseLinkLocalIP

If this is enabled CorePlus will use a Link Local IP (169.254.\*.\*) instead of 0.0.0.0 while waiting for a lease.

Default: *Disabled*

### DHCP\_DisableArpOnOffer

Disable the arp check done by CorePlus on the offered IP.

Default: *Disabled*

## 12.13. DHCPRelay Settings

### DHCPRelay\_MaxTransactions

Maximum number of transactions at the same time.

Default: 32

### DHCPRelay\_TransactionTimeout

For how long a dhcp transaction can take place.

Default: 10 seconds

### DHCPRelay\_MaxPPMPerIface

How many dhcp-packets a client can send to through CorePlus to the dhcp-server during one minute.

Default: 500 packets

### DHCPRelay\_MaxHops

How many hops the dhcp-request can take between the client and the dhcp-server.

Default: 5

### DHCPRelay\_MaxLeaseTime

The maximum lease time allowed through CorePlus, if the DHCP server have higher leases this value will be shorted down to this value.

Default: 10000 seconds

### DHCPRelay\_MaxAutoRoutes

How many relays that can be active at the same time.

Default: 256

### DHCPServer\_SaveRelayPolicy

What policy should be used to save the relay list to the disk, possible settings are Disabled, ReconfShut, or ReconfShutTimer.

Default: *ReconfShut*

### DHCPRelay\_AutoSaveRelayInterval

How often should the relay list be saved to disk if DHCPServer\_SaveRelayPolicy is set to ReconfShutTimer.

Default: 86400

## 12.14. DHCP Server Settings

### DHCP Server \_SaveLeasePolicy

What policy should be used to save the lease database to the disk, possible settings are Disabled, ReconfShut, or ReconfShutTimer.

Default: *ReconfShut*

### DHCP Server \_AutoSaveLeaseInterval

How often should the leases database be saved to disk if DHCP Server \_SaveLeasePolicy is set to ReconfShutTimer.

Default: *86400*

## 12.15. IPsec Settings

### IKESendInitialContact

Determines whether or not IKE should send the "Initial Contact" notification message. This message is sent to each remote gateway when a connection is opened to it and there are no previous IPsec SA using that gateway.

Default: *Enabled*

### IKESendCRLs

Dictates whether or not CRLs (Certificate Revocation Lists) should be sent as part of the IKE exchange. Should typically be set to ENABLE except where the remote peer does not understand CRL payloads.

Default: *Enabled*

### IKECRLValidityTime

A CRL contains a "next update" field that dictates the time and date when a new CRL will be available for download from the CA. The time between CRL updates can be anything from a few hours and upwards, depending on how the CA is configured. Most CA software allow the CA administrator to issue new CRLs at any time, so even if the "next update" field says that a new CRL is available in 12 hours, there may already be a new CRL for download.

This setting limits the time a CRL is considered valid. A new CRL is downloaded when IKECRLValidityTime expires or when the "next update" time occurs. Whichever happens first.

Default: *90000*

### IKEMaxCAPath

When the signature of a user certificate is verified, CorePlus looks at the 'issuer name' field in the user certificate to find the CA certificate the certificate was signed by. The CA certificate may in turn be signed by another CA, which may be signed by another CA, and so on. Each certificate will be verified until one that has been marked trusted is found, or until it is determined that none of the certificates were trusted.

If there are more certificates in this path than what this setting specifies, the user certificate will be considered invalid.

Default: *15*

### IPsecCertCacheMaxCerts

Maximum number of certificates/CRLs that can be held in the internal certificate cache. When the certificate cache is full, entries will be removed according to an LRU (Least Recently Used) algorithm.

Default: *1024*

### IPsecBeforeRules

Pass IKE & IPsec (ESP/AH) traffic sent to CorePlus directly to the IPsec engine without consulting the ruleset.

Default: *Enabled*

## 12.16. Transparent Mode Settings

### Transp\_CAMToL3CDestLearning

Enable this if the gateway should be able to learn the destination for hosts by combining destination address information and information found in the CAM table.

Default: *Enabled*

### Transp\_DecrementTTL

Enable this if the TTL should be decremented each time a packet traverses the gateway in Transparent Mode.

Default: *Disabled*

### Transparency\_CAMSize

This setting can be used to manually configure the size of the CAM table. Normally *Dynamic* is the preferred value to use.

Default: *Dynamic*

### Transparency\_L3CSize

This setting can be used to manually configure the size of the Layer 3 Cache. Normally *Dynamic* is the preferred value to use.

Default: *Dynamic*

### NullEnetSender

Defines what to do when receiving a packet that has the sender hardware (MAC) address in ethernet header set to null (0000:0000:0000). Options:

- *Drop* - Drop packets
- *DropLog* - Drop and log packets

Default: *DropLog*

### BroadcastEnetSender

Defines what to do when receiving a packet that has the sender hardware (MAC) address in ethernet header set to the broadcast ethernet address (FFFF:FFFF:FFFF). Options:

- *Accept* - Accept packet
- *AcceptLog* - Accept packet and log
- *Rewrite* - Rewrite to the MAC of the forwarding interface
- *RewriteLog* - Rewrite to the MAC of the forwarding interface and log
- *Drop* - Drop packets

- *DropLog* - Drop and log packets

Default: *DropLog*

## MulticastEnetSender

Defines what to do when receiving a packet that has the sender hardware (MAC) address in ethernet header set to a multicast ethernet address. Options:

- *Accept* - Accept packet
- *AcceptLog* - Accept packet and log
- *Rewrite* - Rewrite to the MAC of the forwarding interface
- *RewriteLog* - Rewrite to the MAC of the forwarding interface and log
- *Drop* - Drop packets
- *DropLog* - Drop and log packets

Default: *DropLog*

## Transparency\_ATSExpire

Defines the lifetime of an unanswered ARP Transaction State (ATS) entry in seconds. Valid values are 1-60 seconds.

Default: *3 seconds*

## Transparency\_ATSSize

Defines the maximum total number of ARP Transaction State (ATS) entries. Valid values are 128-65536 entries.

Default: *4096*



### **Note**

*Both Transparency\_ATSExpire and Transparency\_ATSSize can be used to tweak the ATS handling to be optimal in different environments.*

## RelaySTP

When set to Ignore, all incoming STP, RSTP and MSTP BPDUs are relayed to all transparent interfaces in the same routing table, except the incoming interface. Options:

- Ignore
- *Log* - Let the packets pass and log the event
- *Drop* - Drop the packets
- *DropLog* - Drop packets log the event

Default: *Drop*

## 12.17. Logging Settings

### LogSendPerSecLimit

This setting limits how many log packets CorePlus may send out per second. This value should never be set too low, as this may result in important events not being logged, nor should it be set too high. One situation where setting too high a value may cause damage is when CorePlus sends a log message to a server whose log receiver is not active. The server will send back an ICMP UNREACHABLE message, which may cause CorePlus to send another log message, which in turn will result in another ICMP UNREACHABLE message, and so on. By limiting the number of log messages CorePlus sends every second, you avoid encountering such devastating bandwidth consuming scenarios.

Default: *3600 seconds, once an hour*



## 12.18. High Availability Settings

### ClusterID

A (locally) unique cluster ID to use in identifying this group of HA Clavister Security Gateways.

Default: *0*

### HASyncBufSize

How much sync data, in KB, to buffer while waiting for acknowledgments from the cluster peer.

Default: *1024*

### HASyncMaxPktBurst

The maximum number of state sync packets to send in a burst.

Default: *20*

### HAInitialSilence

The time to stay silent on startup or after reconfiguration.

Default: *5*

### RFO\_GratuitousARPOnFail

Send gratuitous ARP on failover to alert hosts about changed interface ethernet and IP addresses.  
Possible values: TRUE or FALSE.

Default: *TRUE*

## 12.19. Time Synchronization Settings

### TimeSync\_SyncInterval

Seconds between each resynchronization.

Default: *86400*

### TimeSync\_MaxAdjust

Maximum time drift that a server is allowed to adjust.

Default: *3600*

### TimeSync\_ServerType

Type of server for time synchronization, UDPTIME or SNTP (Simple Network Time Protocol).

Default: *SNTP*

### TimeSync\_GroupIntervalSize

Interval according to which server responses will be grouped.

Default: *10*

### TimeSync\_TimeServerIP1

DNS hostname or IP Address of Timeserver 1.

Default: *none*

### TimeSync\_TimeServerIP2

DNS hostname or IP Address of Timeserver 2.

Default: *none*

### TimeSync\_TimeServerIP3

DNS hostname or IP Address of Timeserver 3.

Default: *none*

### TimeSync\_TimeZoneOffs

Time zone offset in minutes.

Default: *0*

### TimeSync\_DSTEnabled

Perform DST adjustment according to DSTOffs/DSTStartDate/DSTEndDate.

Default: *OFF*

## TimeSync\_DSTOffs

DST offset in minutes.

Default: *0*

## TimeSync\_DSTStartDate

What month and day DST starts, in the format MM-DD.

Default: *none*

## TimeSync\_DSTEndDate

What month and day DST ends, in the format MM-DD.

Default: *none*

## 12.20. DNS Client Settings

### **DNS\_DNSServerIP1**

Primary DNS Server.

Default: *none*

### **DNS\_DNSServerIP2**

Secondary DNS Server.

Default: *none*

### **DNS\_DNSServerIP3**

Tertiary DNS Server.

Default: *none*

## 12.21. Remote Administration Settings

### NetConBiDirTimeout

When uploading a new configuration, CorePlus tries to establish bi-directional communication back to Clavister FineTune in order to guarantee Clavister Security Gateway reachability. This setting specifies how long to wait before CorePlus reverts to the previous configuration.

Default: *30 seconds*

### NetconBeforeRules

For netcon traffic to CorePlus: look in the Remotes section before consulting the ruleset.

Default: *Enabled*

### SNMPBeforeRules

For SNMP traffic to CorePlus: look in the Remotes section before consulting the ruleset.

## 12.22. HTTP Poster Settings

### HTTPPoster\_URL1, HTTPPoster\_URL2, HTTPPoster\_URL3

The URLs specified here will be posted in order when CorePlus is loaded.

### HTTPPoster\_RepDelay

Delays in seconds until all URLs are refetched.

Default: *604800*

## 12.23. Hardware Performance Settings

For more information in how to change Hardware Performance settings, see the Knowledge Base Article #10024. -->

### **Ringsize\_e1000\_rx**

Size of the rx buffer on e1000 cards.

Default: *64*

### **Ringsize\_e1000\_tx**

Size of the tx buffer on e1000 cards.

Default: *256*

### **Ringsize\_e100\_rx**

Size of the rx buffer on e100 cards.

Default: *32*

### **Ringsize\_e100\_tx**

Size of the tx buffer on e100 cards.

Default: *128*

## 12.24. PPP Settings

### PPP\_L2TPBeforeRules

Pass L2TP traffic sent to the Clavister Security Gateway directly to the L2TP Server without consulting the ruleset.

Default: *Enabled*

### PPP\_PPTPBeforeRules

Pass PPTP traffic sent to the Clavister Security Gateway directly to the PPTP Server without consulting the ruleset.

Default: *Enabled*



## 12.25. RADIUS Accounting Settings

### AllowAuthIfNoAccountingResponse

If there is no response from a configured RADIUS accounting server when sending accounting data for a user that has already been authenticated, then enabling this settings means that the user will continue to be logged in.

Disabling the setting will mean that the user will be logged out if the RADIUS accounting server cannot be reached even though the user has been previously authenticated.

Default: *Enabled*

### LogOutAccUsersAtShutdown

If there is an orderly shutdown of the Clavister Security Gateway by the administrator, then CorePlus will delay the shutdown until it has sent RADIUS accounting **STOP** messages to any configured RADIUS server.

If this option is not enabled, CorePlus will shutdown even though there may be RADIUS accounting sessions that have not be correctly terminated. This could lead to the situation that the RADIUS server will assume users are still logged in even though their sessions have been terminated.

Default: *Enabled*

## 12.26. IDP

### IDP\_UpdateInterval

The number of seconds between automatic IDP signature updates. A value of 0 stops automatic updates.

Default: *43200 (=12 hours)*

## 12.27. Multicast Settings

### AutoAddMulticastCoreRoute

This setting will automatically add core routes in all routing tables for the multicast IP address range 224.0.0.0/4. If the setting is disabled, multicast packets might be forwarded according to the default route.

Default: *Enabled*

### IGMPBefore Rules

For IGMP traffic, by pass the normal IP rule set and consult the IGMP rule set.

Default: *Enabled*

### IGMPMaxReqs

The maximum global number of IGMP messages to process each second.

Default: *1000*

### IGMPRouterVersion

The IGMP protocol version that will be globally used on interfaces without a configured IGMP Setting. Multiple querying IGMP routers on the same network must use the same IGMP version. Global setting on interfaces without an overriding IGMP Setting.

Default: *IGMPv3*

### IGMPLowestCompatibleVersion

IGMP messages with a version lower than this will be logged and ignored. Global setting on interfaces without an overriding IGMP Setting.

Default: *IGMPv1*

### IGMPMaxReqsIf

The maximum number of requests per interface and second. Global setting on interfaces without an overriding IGMP Setting.

Default: *100*

### IGMPreactToOwnQueries

The security gateway should always respond with IGMP Membership Reports, even to queries originating from itself. Global setting on interfaces without an overriding IGMP Setting.

Default: *Disabled*

### IGMPRobustnessVariable

IGMP is robust to (IGMPRobustnessVariable - 1) packet losses. Global setting on interfaces without an overriding IGMP Setting.

Default: *2*

## IGMPQueryInterval

The interval (ms) between General Queries sent by the device to refresh its IGMP state. Global setting on interfaces without an overriding IGMP Setting.

Default: *125,000 ms*

## IGMPQueryResponseInterval

The maximum time (ms) until a host has to send a reply to a query. Global setting on interfaces without an overriding IGMP Setting.

Default: *10,000 ms*

## IGMPStartupQueryInterval

The interval of General Queries (ms) used during the startup phase. Global setting on interfaces without an overriding IGMP Setting.

Default: *30,000 ms*

## IGMPStartupQueryCount

The security gateway will send IGMPStartupQueryCount general queries with an interval of IGMPStartupQueryInterval at startup. Global setting on interfaces without an overriding IGMP Setting.

Default: *2*

## IGMPLastMemberQueryInterval

The maximum time (ms) until a host has to send an answer to a group or group-and-source specific query. Global setting on interfaces without an overriding IGMP Setting.

Default: *5,000 ms*

## IGMPUnsolicitedReportInterval

The time (ms) between repetitions of an initial membership report. Global setting on interfaces without an overriding IGMP Setting.

Default: *1,000 ms*

## 12.28. Hardware Monitor Settings

### HWM\_PollInterval

Polling interval for Hardware Monitor which is the delay in milliseconds between reading of hardware monitor values. Minimum 100, Maximum 10000.

Default: *500 ms*

### HWMMem\_Interval

Memory polling interval which is the delay in minutes between reading of memory values. Minimum 1, Maximum 200.

Default: *15 mins*

### HWMMem\_LogRepetition

Should we send a log message for each poll result that is in the Alert, Critical or Warning level, or should we only send when a new level is reached. If True, a message is sent each time HWM-Mem\_Interval is triggered. If False, a message is sent when a value goes from one level to another.

Default: *False*

### HWMMem\_UsePercent

True if the memory monitor uses a percentage as the unit for monitoring, False if it uses Megabyte. Applies to HWMMem\_AlertLevel, HWMMem\_CriticalLevel and HWMMem\_WarningLevel.

Default: *True*

### HWMMem\_AlertLevel

Generate an Alert log message if free memory is below this value. Disable by setting to 0. Maximum value is 10,000.

Default: *0*

### HWMMem\_CriticalLevel

Generate a Critical log message if free memory is below this value. Disable by setting to 0. Maximum value is 10,000.

Default: *0*

### HWMMem\_WarningLevel

Generate a Warning log message if free memory is below this value. Disable by setting to 0. Maximum value 10,000.

Default: *0*

### AlarmRepeatInterval

The delay in seconds between alarms when a continuous alarm is used. Minimum 0, Maximum 10,000.

Default: *60 seconds*

## 12.29. Packet Re-assembly Settings

Packet re-assembly collects IP fragments into complete IP datagrams and, for TCP, reorders segments so that they are processed in the correct order and also to keep track of potential segment overlaps and to inform other sub-systems of such overlaps. The associated settings limit memory used by the re-assembly sub-system.

### Reassembly\_MaxConnections

This setting specifies how many connections can use the re-assembly system at the same time. It is expressed as a percentage of the total number of allowed connections. Minimum 1, Maximum 100.

Default: *80*

### Reassembly\_MaxProcessingMem

This setting specifies how much memory that the re-assembly system can allocate to process packets. It is expressed as a percentage of the total memory available. Minimum 1, Maximum 100.

Default: *3*

## 12.30. Miscellaneous Settings

### BufFloodRebootTime

As a final way out, CorePlus automatically reboots if it's buffers have been flooded for a long time. This setting specifies this amount of time.

Default: *3600*

### ScrSaveTime

The time in seconds before CorePlus automatically enables its screen saver. The screen saver will automatically adapt its activity to the current CPU load of the system. During high loads, it will update once per second, consuming a fraction of a percent of CPU load.

Default: *300 seconds (5 minutes)*

### HighBuffers

The number of buffers to allocate in RAM above the 1 MB limit.

Default: *3% of total RAM, with a lower limit of 1024+lowbuffers*

### MaxPipeUsers

The maximum number of pipe users to allocate. As pipe users are only tracked for a 20th of a second, this number usually does not need to be anywhere near the number of actual users, or the number of statefully tracked connections. If there are no configured pipes, no pipe users will be allocated, regardless of this settings. For more information about pipes and pipe users, see chapter 10, Traffic Shaping.

Default: *512*





---

# Appendix A. Subscribing to Security Updates

## Introduction

The CorePlus Intrusion Detection and Prevention (IDP) module, the Anti-Virus module, as well as the Web Content Filtering Module, function by accessing the Clavister Service Provisioning Network. The following 3 CorePlus modules make use of this network:

- **Intrusion Detection and Prevention (IDP)** - A database of known threat signatures is stored locally in the Clavister Security Gateway. This local database needs to be updated regularly with the latest threat signatures by automatically downloading updates from the Clavister Service Provisioning Network.
- **Anti-Virus Scanning** - Like IDP, a database of known virus signatures is stored locally. This local database also needs to be updated regularly with the latest virus signatures by automatically downloading updates from the Clavister Service Provisioning Network.
- **Web Content Filtering** - accesses the database available on the Clavister Service Provisioning Network for each website accessed in order to categorize them and implement the chosen filtering policy. CorePlus caches already looked-up websites locally to maximise look-up performance.

To make use of any or all of these CorePlus features, a subscription for each one must be taken out. This an addition to the standard CorePlus license. This is done by:

1. Purchasing the required subscription from your Clavister reseller. A subscription can be purchased with a lifetime of 1 year or 3 years. Each subscription is recorded in the central Clavister customer license database and becomes associated with your overall Clavister license.
2. Once your purchase is logged, use the clientweb or use Clavister FineTune to update your Clavister Security Gateway license. This will cause the central Clavister database to be checked and your local Clavister Security Gateway license will be updated with the subscription information.



### ***Important***

*Make sure that the DNS address is set correctly in Advanced Settings so the Clavister Provisioning Network servers can be located by CorePlus.*

## Subscription renewal

When any subscription is approaching its expiry date, you will receive notification in 3 ways:

- Clavister FineTune will bring up a reminder dialog.
- A reminder email will be sent by Clavister to the email address associated with the license.
- Providing a log server has been configured, a log message will be sent which indicates subscription renewal is required.



### ***Renew the subscription in good time!***

*Make sure a subscription is renewed well before the current subscription ends.*

# IDP and Anti-Virus Database Updating

These services operate by downloading "signature" patterns which are used by CorePlus to search for the most recently recognised security threats in internet traffic as well as viruses in downloads.

New threats are being identified every day and the signature databases in these modules need to be updated regularly. A subscription means that CorePlus will periodically access a central server and update the copy of the database on the local Clavister Security Gateway with the latest signatures. Database updates can involve as many as 20 signature changes or more in a single day.

By default CorePlus will check for updates every 12 hours. The frequency of checking for updates can be set explicitly through **Advanced Settings**. It can be set to zero if updates are not to be done automatically.

## Database Console Commands

IDP and Anti-Virus (AV) databases can be controlled directly through a number of console commands.

### Pre-empting Database Updates

An IDP database update can be forced at any time by using the command:

```
Cmd> updatecenter -update idp
```

An Anti-Virus update can similarly be initiated with the command:

```
Cmd> updatecenter -update av
```

### Querying Update Status

To get the status of IDP updates use the command:

```
Cmd> updatecenter -status idp
```

To get the status of AV updates:

```
Cmd> updatecenter -status av
```

### Querying Server Status

To get the status of the Clavister network servers use the command:

```
Cmd> updatecenter -servers
```

### Deleting Local Databases

Some technical problem in the operation of either IDP or the Anti-Virus modules may be resolved by deleting the database and reloading. For IDP this is done with the command

```
Cmd> removedb idp
```

To remove the Anti-Virus database, use the command:

```
Cmd> removedb av
```

Once removed, the entire system should be rebooted and a database update initiated. Removing the database is also recommended if either IDP or Anti-Virus is not used for longer periods of time.

**Note**

*Anti-Virus database updates may require a couple of seconds to be optimized once an update is downloaded. This will cause the gateway to sometimes momentarily pause in its operation. This behaviour is normal. Deleting the databases can cause a similar pause.*

---

# Appendix B. CLI Reference

## About

Brings up information pertaining to the version of the gateway core in use and a copyright notice.

**Syntax:** about

*Example*

```
Cmd> about
Clavister Security Gateway 8.50.00V
Copyright Clavister 2006. All rights reserved
SSH IPSEC Express SSHIPM version 5.1.1 library 5.1.1
Copyright 1997-2003 SSH Communications Security Ltd.
Build : Sep 22 2003
```

## Access

Displays the contents of the Access configuration section.

**Syntax:** access

*Example*

```
Cmd> access
Source IP Address Access list (spoofing protection)
Default action is DROP
Symbolic Name   Action   Iface   Source Range
-----
ExpectIntnet    Expect   lan     192.168.123.0/24
ExpectExtnet    Expect   wan     194.2.1.0/24
ExpectWorld     Expect   wan     0.0.0.0/0
```

## ARP

Displays ARP entries for the specified interface(s). Published, static as well as dynamic items are shown.

**Syntax:** arp [options] <interface pattern>

Options:

- ip <pattern> - Display only IP addresses matching <pattern>
- hw <pattern> - Display only hardware addresses matching <pattern>
- num <n> - Display only the first <n> entries per iface (default: 20)
- hashinfo - Display information on hash table health
- flush - Flush ARP cache of ALL interfaces
- flushif - Flush ARP cache of an iface

*Example*

```
Cmd>arp wan
ARP cache of iface wan
Dynamic 194.2.1.1 = 0020:d216:5eec Expire=141
```

## ARPSnoop

Toggles the on-screen display of ARP queries. This command can be of great help in configuring the hardware, since it shows which IP addresses are heard on each interface.

**Syntax:** arpsnoop <interface pattern>

Toggle snooping on given interfaces.

**Syntax:** arpsnoop all

Snoop all interfaces.

**Syntax:** arpsnoop none

Disable all snooping.

*Example*

```
Cmd> arpsnoop all
ARP snooping active on interfaces: lan wan dmz
ARP on wan: gw-world requesting ip_wan
ARP on lan: 192.168.123.5 requesting ip_lan
```

## Buffers

This command can be useful in troubleshooting; e.g. if an unexpectedly large number of packets begin queuing or when traffic does not seem to be flowing for some inexplicable reason. By analyzing the contents of the buffers, it is possible to determine whether such traffic is making it to the Clavister Security Gateway at all.

**Syntax:** buffers

Brings up a list of most recently freed buffers.

*Example*

```
Cmd> buff
Displaying the 20 most recently freed buffers
```

RecvIf	Num	Size	Protocol	Sender	Destination
wan	1224	121	UDP	192.168.3.183	192.168.123.137
lan	837	131	UDP	192.168.123.137	192.168.3.183
wan	474	112	UDP	192.168.3.183	192.168.123.137
wan	395	91	UDP	192.168.3.183	192.168.123.137
lan	419	142	UDP	192.168.123.137	192.168.3.183
wan	543	322	UDP	194.2.1.50	192.168.123.182
lan	962	60	UDP	192.168.123.182	194.2.1.50
lan	687	60	ARP	0080:ad87:e592	ffff:ffff:ffff
wan	268	88	UDP	192.168.3.183	192.168.123.137
lan	249	101	UDP	192.168.123.137	192.168.3.183
wan	219	60	TCP	193.12.33.105	192.168.123.12
lan	647	60	ARP	0010:a707:dd31	ffff:ffff:ffff
wan	1185	98	UDP	192.168.3.183	192.168.123.137
lan	912	98	UDP	192.168.123.137	192.168.3.183
wan	682	112	UDP	192.168.3.183	192.168.123.137
lan	544	60	TCP	192.168.123.12	194.2.1.50
lan	633	60	TCP	192.168.123.26	194.2.1.50
lan	447	60	TCP	192.168.123.25	194.2.1.50
lan	645	60	TCP	192.168.123.23	194.2.1.50
lan	643	123	UDP	192.168.123.137	192.168.3.183

**Syntax:** buffer <number>

Shows the contents of the specified buffer.

*Example*

```
Cmd> buff 1059
Decode of buffer number 1059
lan: Enet 0050:dadf:7bbf > 0003:325c:cc00 type 0x0800 len 1058
IP 192.168.123.10 -> 193.13.79.1 IHL:20
DataLen:1024 TTL:254 Proto:ICMP
ICMP Echo reply ID:6666 Seq:0
```

**Syntax:** buffer

Shows the contents of the most recently used buffer.

*Example*

```
Cmd> buff .
Decode of buffer number 1059
lan: Enet 0050:dadf:7bbf > 0003:325c:cc00 type 0x0800 len 1058
IP 192.168.123.10 -> 193.13.79.1 IHL:20
DataLen:1024 TTL:254 Proto:ICMP
ICMP Echo reply ID:6666 Seq:0
```

## Certcache

Displays the contents of the certificate cache.

**Syntax:** certcache

## CfgLog

Shows the results of the most recent reconfiguration or start up of the gateway. This text is the same as is shown on-screen during reconfiguration or start up.

**Syntax:** cfglog

*Example*

```
Cmd> cfglog
Configuration log:
Configuring from FWCore_N.cfg
Configuration done
Configuration "FWCore_N.cfg" (v153)
verified for bi-directional communication
```

## Connections

Shows the last 20 connections opened through the Clavister Security Gateway. Connections are created when traffic is permitted to pass via Allow or NAT rules. Traffic permitted to pass under Fwd-Fast is not included in this list. Each connection has two timeout values, one in each direction. These are updated when the gateway receives packets from each end of the connection. The value shown in the Timeout column is the lower of the two values.

Possible values in the State column include:  
SYN\_RECV TCP packet with SYN flag received  
SYNACK\_S TCP packet with SYN + ACK flags sent  
ACK\_RECV TCP packet with ACK flag received  
TCP\_OPEN TCP packet with ACK flag sent

FIN\_RECV TCP packet with FIN / RST flag received  
PING The connection is an ICMP ECHO connection  
UDP The connection is a UDP connection  
RAWIP The connection uses an IP protocol other than TCP, UDP or ICMP

**Syntax:** connections

*Example*

```
Cmd> conn
State Prot Source Destination Time
TCP_OPEN TCP wan:60.20.37.6:5432 dmz:wwwsrv:80 3600
SYN_RECV TCP wan:60.20.37.6:5433 dmz:wwwsrv:80 30
UDP_OPEN UDP lan:10.5.3.2:5433 dmz:dnsrv:53 50
```

## Cpuid

Shows information regarding the CPU in the Clavister Security Gateway.

**Syntax:** cpuid

*Example*

```
Cmd> cpuid
Processor: Intel Pentium III, III Xeon, or Celeron
Brand ID: Intel Pentium III
Frequency: 996 Mhz
Family: 6
Model: 8
Stepping: 10
Vendor id: GenuineIntel
Type: Original OEM Processor
Feature flags:
fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse-36 psn mmx fxsr sse
Cache and TLB information:
0x01: Instruction TLB, 4K pages, 4-way set associative, 32 entries
0x02: Instruction TLB, 4M pages, fully associative, 2 entries
0x03: Data TLB, 4K pages, 4-way set associative, 64 entries
0x08: Instruction cache 16K, 4-way set associative, 32 byte line
0x04: Data TLB, 4M pages, 4-way set associative, 8 entries
0x0C: Data cache, 16K, 4-way set associative, 32 byte line size
```

## DHCP

**Syntax:** dhcp [options] <interface>

Options:

-renew - Force interface to renew it's lease  
-release - Force interface to release it's lease

*Example*

```
Cmd> dhcp -renew wan
```

## DHCPRelay

Show the contents of the DHCP-relay configuration section.

**Syntax:** dhcprelay [options]



## Options:

-release ip - Releases the IP and removes associated routes from the Clavister Security Gateway.

*Example*

```

Cmd> dhcprelay
Content of the DHCP/BOOTP-relayer ruleset; default action is IGNORE
# Act. Source HW-Filter Server Allowed
1 RELAY vlan1,vlan2.. * 192.168.0.10 192.168.0.100 - 192.168.0.120

Dynamically added routes for relayed DHCP leases
Iface Host-Route Local IP ProxyARP Expire
vlan2 192.168.0.101/32 internals 3539
vlan4 192.168.0.112/32 internals 3539

```

## DHCP Server

Show the contents of the DHCP-server configuration section and active DHCP leases.

**Syntax:** dhcpserver [options]

## Options:

-rules - Shows dhcp server rules  
 -leases - Shows dhcp server leases  
 -mappings - Shows dhcp server IP=>MAC mappings  
 -release - Releases an active or blacklisted IP

*Example*

```

Cmd> dhcpserver
Contents of the DHCP-Server ruleset; default action is IGNORE
# Source Pool Gateway DNS1 LTime
-----
1 lan 192.168.32.1-.39.1, ... 192.168.39.254 192.168.39.253 10800

Active DHCP sessions:
Rule Iface Client MAC Client IP Expire
-----
1 lan 000f:3d0f:797a 192.168.32.212 10746
1 lan 0050:8df5:24a3 192.168.37.88 10700
1 lan 000f:3d3f:c409 192.168.34.96 10678
1 lan 000d:8802:61cd 192.168.34.175 10574
1 lan 000f:3d1f:a3cc 192.168.34.154 10549
1 lan 0030:f12e:587f 192.168.38.220 10529

```

## DynRoute

Displays the dynamic routing policy filter ruleset and current exports.

**Syntax:** dynroute [options]

## Options:

-rules - Shows the dynamic policy filter ruleset  
 -exports - Displays current exports

## Fragments

Shows the 20 most recent fragment reassembly attempts. This includes both ongoing and completed attempts.

**Syntax:** frags

*Example*

```
Cmd> frags
RecvIf Num State Source Destination Proto Next Timeout
lan 2 Done 10.5.3.2 26.23.5.4 ICMP 2000 58
wan 8 Accept 23.3.8.4 10.5.3.2 ICMP 1480 60
```

## HA

Shows information about a HA cluster.

*Example*

```
Cmd> ha
This device is a HA SLAVE
This device is currently ACTIVE (will forward traffic)
HA cluster peer is ALIVE
```

## HTTPPoster

Show the configured httpposter urls and status.

**Syntax:** httpposter [options]

Options:

-repost - Re-post all URLs now.

*Example*

```
Cmd> httpposter
HTTPPoster_URL1:
Host : ""
Port : 0
Path : ""
Post : ""
User : ""
Pass : ""
Status: (not configured)

HTTPPoster_URL2:
Host : ""
Port : 0
Path : ""
Post : ""
User : ""
Pass : ""
Status: (not configured)

HTTPPoster_URL3:
Host : ""
Port : 0
Path : ""
Post : ""
User : ""
Pass : ""
Status: (not configured)
```

## Ifacegroups

Shows the configured interface groups.

**Syntax:** ifacegroups <name pattern>

*Example*

```
Cmd> ifacegroups
Configured interface groups:
internals lan,vlan1,vlan2,vlan3
```

## IfStat

**Syntax:** ifstat

Shows a list of the interfaces installed.

*Example*

```
Cmd> ifstat
Configured interfaces:
Iface IP Address      PBR membership  Interface type
-----
core 127.0.0.1         <all>          Null (sink)
mgmt 10.9.0.36          <all>          Builtin e100 - Intel(R) 8255..
wan 172.16.87.252     <all>          Builtin e100 - Intel(R) 8255..
lan 192.168.121.1     <all>          Builtin e100 - Intel(R) 8255..
pptp 10.10.240.131     <all>          PPTP tunnel to 192.168.23.1
```

**Syntax:** ifstat <interface>

Shows hardware and software statistics for the specified NIC.

*Example*

```
Cmd> ifstat lan
Iface lan
Builtin e1000 - Intel(R) PRO/1000 T Server Adapter Slot 2/1 IRQ 5
Media : "1000BaseTx"
Speed : 1000 Mbps Full Duplex
MTU : 1500
Link Partner :
10BASE-T, 10BASE-T FD, 100BASE-TX, 100BASE-TX FD, 1000BASE-TX F
Bus Type : PCI 64-bit/33MHz
IP Address : 192.168.123.1
Hw Address : 0003:47ab:ea25

Software Statistics:
Soft received : 193075 Soft sent : 212480 Send failures : 0
Dropped : 0 IP Input Errs : 0

Hardware statistics:
IN : packets= 193074 bytes=36524718 errors= 10 dropped= 10
OUT: packets= 212646 bytes=208065794 errors= 0 dropped= 0
Collisions : 0
In : Length Errors : 0
In : Overruns : 0
In : CRC Errors : 0
In : Frame Errors : 0
In : FIFO Overruns : 0
In : Packets Missed : 0
Out: Sends Aborted : 0
Out: Carrier Errors : 0
Out: FIFO Underruns : 0
Out: SQE Errors : 0
```

```
Out: Late Collisions : 0
```

The Dropped counter in the software section states the number of packets discarded as the result of structural integrity tests or rule-set drops. The IP Input Errs counter in the software section specifies the number of packets discarded due to checksum errors or IP headers broken beyond recognition. The latter is most likely the result of local network problems rather than remote attacks.

## Ikesnoop

Ikesnoop is used to diagnose problems with IPsec tunnels.

**Syntax:** ikesnoop

Display current ikesnoop status.

**Syntax:** ikesnoop off

Turn IKE snooping off.

**Syntax:** ikesnoop on [ipaddr]

Turn IKE snooping on, if an IP is specified then only IKE traffic from that IP will be shown.

**Syntax:** ikesnoop verbose [ipaddr]

Enable verbose output, if an IP is specified then only IKE traffic from that IP will be shown.

## Ipseckeealive

Show the status of the configured IPsec keepalive connections.

*Example*

```
Cmd> ipseckeealive
192.168.0.10->192.168.1.10: Consecutive lost: 0 sent: 908 lost: 2
192.168.1.10->192.168.0.10: Consecutive lost: 0 sent: 913 lost: 6
```

## IPSec tunnels

Display configured IPsec VPN connections.

**Syntax:** ipsectunnels

*Example*

```
Cmd> ipsectunnel
VPN Conns list
No Name Local Net Remote Net Remote GW
0 vpn-home 192.168.123.0/24 0.0.0.0/0 None
MAIN_MODE SA_PER_NET DONT_VERIFY_PAD IKE group: 2
IKE proplist: ike-default, IPsec proplist: esp-tn-roamingclients
```

## IPSecstats

Display connected IPsec VPN gateways and remote clients.

**Syntax:** ipsecstats <options>

Options: -u  
Append SA usage  
-num <n>

*Example*

```
Cmd> ipsecstats
--- IPsec SAs:
Displaying one line per SA-bundle
VPN Tunnel Local net Remote net Remote GW
-----
vpn-home 192.168.123.0/24 192.168.1.2/32 192.168.1.2/32
```

## Killsa

Kills all IPsec and IKE SAs for the specified IP-address.

**Syntax:** killsa <ipaddr>

*Example*

```
Cmd> killsa 192.168.0.2
Destroying all IPsec & IKE SAs for remote peer 192.168.0.2
```

## License

Shows the content of the license-file. It is also possible to remove a license from a running system with this command, by doing a license remove.

**Syntax:** license [remove]

*Example*

```
Cmd> lic
Contents of the License file
-----
Registration key: 1234-1234-1234-1234
Bound to MAC address: 00-48-54-00-3b-00
Company: Clavister AB
Registration date: 2002-11-20 00:00:00.000
Issued date: 2002-11-20 19:22:38
Last modified: 2002-11-20 19:22:27
New upgrades until: 2003-02-14 00:00:00.000
Ethernet Interfaces: 4
Max Connections: 128000
Max Routes: 2048
Max Rules: 16000
Max Throughput: 200
Max VPN Tunnels: 20
Max VLANs: 2048
```

## Lockdown

Sets local lockdown on or off. During local lockdown, only traffic from admin nets to CorePlus itself is allowed. Everything else is dropped. Note: If local lockdown has been set by the core itself due to licensing / configuration problems, this command will NOT remove such a lock.

**Syntax:** lockdown [ on | off ]

## Loghosts

Shows the list of log recipients CorePlus is configured to send log data to.

**Syntax:** loghosts

*Example*

```
Cmd> loghosts
Log hosts:
SysLog 192.168.123.10 Facility: local0
Usage logging in 3600 second intervals
```

## Logout

Only works on the serial or local console, it is used to logout the current user and enable the password.

## Memory

Displays core memory consumption. Also displays detailed memory use of some components and lists.

**Syntax:** memory

## Netcon

Shows a list of users currently connected via the netcon protocol.

**Syntax:** netcon

*Example*

```
Cmd> netcon
Currently connected NetCon users:
Iface IP address port
lan 192.168.123.11 39495
```

## Netobjects

Shows the contents of the Nets configuration section. If a Netobject is specified the output will show user authentication information associated with that object.

**Syntax:** netobjects [netobject]

*Example*

```
Cmd> netobjects
List of named network objects:
wannet 194.2.1.0/24
lannet 192.168.123.0/24
all-nets 0.0.0.0/0
ip_wan 194.2.1.2/32
br_wan 194.2.1.255/32
br_lan 192.168.123.255/32
ip_lan 192.168.123.1/32
gw-world 194.2.1.1/32
```

## OSPF

Shows runtime information about the OSPF router processes) and is used to stop/start OSPF processes).

**Syntax:** ospf [<process name>] [<parameter>] [<argument>]

NOTE: <process name> is only required if there are more than one OSPF routing process

Options:

iface [<iface>] - Display interface information  
 area [<areaID>] - Display area information  
 neighbor [<if>:][<neiID>] - Display neighbor information  
 route, Display the internal OSPF process routingtable  
 database [verbose] - Display the LSA database  
 lsa <lsaID> - Display details for a specified LSA  
 snoop [on|off] - Display troubleshooting messages on the console  
 ifacedown <iface> - Takes specified interface offline  
 ifaceup <iface> - Takes specified interface online  
 stop - Stop OSPF process  
 start - Start OSPF process  
 restart - Restart OSPF process

## Ping

Sends a specified number of ICMP Echo Request packets to a given destination. All packets are sent in immediate succession rather than one per second. This behavior is the best one suited for diagnosing connectivity problems.

**Syntax:** ping <IPAddr> [<options>] [<# of packets> [<size>]]

Options:

-r <recvif> - Run through the ruleset, simulating that the packet was received by <recvif>  
 -s <srcip> - Use this source IP  
 -p <table> - Route using the specified PBR table  
 -v - Verbose ping

*Example*

```
Cmd> ping 192.168.12.1
Sending 1 ping to 192.168.12.1 from 192.168.14.19
using PBR table "main".
Echo reply from 192.168.12.1 seq=0 time= 10 ms TTL=255

Cmd> ping 192.168.12.1 -v
Sending 1 ping to 192.168.12.1 from 192.168.14.19
using PBR table "main".
... using route "192.168.12.0/22 via wan, no gw" in PBR table "main"
Echo reply from 192.168.12.1 seq=0 time=<10 ms TTL=255
```

## Pipes

Shows the list of configured pipes; the contents of the Pipes configuration section, along with basic throughput figures of each pipe.

**Syntax:** pipes

Displays all configured pipes.

*Example*

```
Cmd> pipes
```

```

Configured pipes:
Name      Grouping  Bits/s Pkts/s Precedence
-----
std-in    Per DestIP          0 1 7
Current:  42.5 K 21.0
std-out   Per SrcIP          0 1 7
Current:  89.1 K 21.0

```

**Syntax:** pipes <name>

Displays in-depth details about the given pipe.

**Syntax:** pipes -u <name>

Displays the 20 currently most active users of the given pipe.

## Proplists

Lists the configured proposal lists.

**Syntax:** proplists [<vpnconn>]

*Example*

```

Cmd> propl
Displaying all configured proposal lists:
ike-default
Type : ISAKMP
Life : 5000KB, 43200s
Cipher : cast128-cbc
Hash : sha1
Type : ISAKMP
Life : 5000KB, 43200s
Cipher : cast128-cbc
Hash : md5
Type : ISAKMP
Life : 5000KB, 43200s
Cipher : 3des-cbc
Hash : sha1
Type : ISAKMP
Life : 5000KB, 43200s
Cipher : 3des-cbc
Hash : md5
esp-tn-lantolan
Type : ESP
Life : 50000KB, 21600s
Cipher : blowfish-cbc
Hmac : hmac-sha1-96
Type : ESP
Life : 50000KB, 21600s
Cipher : blowfish-cbc
Hmac : hmac-md5-96
Type : ESP
Life : 50000KB, 21600s
Cipher : cast128-cbc
Hmac : hmac-sha1-96
Type : ESP
Life : 50000KB, 21600s
Cipher : cast128-cbc
Hmac : hmac-md5-96

```

## ReConfigure



Re-reads the FWCore.cfg file from disk. This process takes approximately one second if done from floppy disk, and approximately a tenth of a second from hard disk or flash disk. If there is a FW-Core\_N.cfg file present on the disk, this will be read instead. However, as there is no Clavister Fine-Tune to attempt two-way communication with CorePlus, it will conclude that the configuration is incorrect and revert to FWCore.cfg after the bi-directional verification timeout period has expired (typically 30 seconds).

**Syntax:** reconfiure

*Example*

```
Cmd>reconfigure
Shutdown RECONFIGURE. Active in 1 seconds.
Shutdown reason: Reconfigure due to CLI command
```

## Remotes

Shows the contents of the Remotes configuration section.

**Syntax:** remotes

*Example*

```
Cmd> remotes
Hosts/nets with remote control of gateway:
lan 192.168.0.12/32 Configure/Update Access
```

## Routes

Displays information about the routing tables, contents of a (named) routing table or a list of routing tables, along with a total count of route entries in each table, as well as how many of the entries are single-host routes. Note that "core" routes for interface IP addresses are not normally shown, use the -all switch to show core routes also. In the "Flags" field of the routing tables, the following letters are used:

O: Learned via OSPF X: Route is Disabled

M: Route is Monitored A: Published via Proxy ARP

D: Dynamic (from e.g. DHCP relay, IPsec, L2TP/PPP servers, etc.)

**Syntax:** routes [<options>] [<table name>]

Options: -all - Also show routes for interface addresses

-num <n> - Limit display to <n> entries (default: 20)

-nonhost - Do not show single-host routes

-tables - Display list of named (PBR) routing tables

-lookup <ip> - Lookup the route for the given IP address

-v - Verbose

*Example*

```
Cmd> routes
Flags Network          Iface  Gateway  Local IP  Metric
-----
      192.168.12.0/22   lan
      194.2.1.0/24     wan
      0.0.0.0/0        wan     194.2.1.1  0
```

```
Cmd> routes -lookup 193.1.2.3
Looking up 193.1.2.3 in routing table "main":
```

```
Matching route: 0.0.0.0/0
Routing table : main
```

```
Send via iface: wan
Gateway : 194.2.1.1

Proxy ARP on :
Local IP : (use iface IP in ARP queries)
Metric : 0
Flags :
```

## Rules

**Syntax:** rules [<options>] [<range>]

Shows the contents of the Rules configuration section.

Options:

- u - Append usage information
- l - Append logging information
- n - Append symbolic rule names
- p - Append pipe information
- a - Append all the information above

The range parameter specifies which rules to include in the output of this command.

*Example*

```
Cmd> rule -u 11-12
contents of ruleset; default action is DROP
Act. Source Destination Protocol/Ports
1 Drop any:0.0.0.0/0 any:0.0.0.0/0 PORTS ALL > 135-139
Use: 0
2 Drop any:0.0.0.0/0 any:0.0.0.0/0 PORTS ALL > 445
Use: 0
```

## Scrsave

Activates the screensaver included with CorePlus.

**Syntax:** scrsave

*Example*

```
Cmd>scr
Activating screen saver...
```

## Services

Displays the list of named services. Services implicitly defined inside rules are not displayed.

**Syntax:** services [<name or wildcard>]

## Settings

Shows the contents of the Settings configuration section.

**Syntax:** settings

Shows available groups of settings.

*Example*

```
Cmd>sett
Available categories in the Settings section:
IP - IP (Internet Protocol) Settings
TCP - TCP (Transmission Control Protocol) Settings
ICMP - ICMP (Internet Control Message Protocol) Settings
ARP - ARP (Address Resolution Protocol) Settings
State - Stateful Inspection Settings
ConnTimeouts - Default Connection timeouts
LengthLim - Default Length limits on Sub-IP Protocols
Frag - Fragmentation Settings
VLAN - VLAN Settings
SNMP - SNMP Settings
DHCP - DHCP (Dynamic Host Configuration Protocol) Settings
Log - Log Settings
Misc - Miscellaneous Settings
```

**Syntax:** settings <group\_name>

Shows the settings of the specified group.

*Example*

```
Cmd> settings arp
ARP (Address Resolution Protocol) Settings
ARPMatchEnetSender : DropLog
ARPQueryNoSenderIP : DropLog
ARPSenderIP : Validate
UnsolicitedARPReplies : DropLog
ARPRequests : Drop
ARPChanges : AcceptLog
StaticARPChanges : DropLog
ARPExpire : 900 ARPExpireUnknown : 15
ARPMulticast : DropLog
ARPBroadcast : DropLog
ARPCacheSize : 4096 ARPHashSize : 512
ARPHashSizeVLAN : 64
```

## Shutdown

Instructs CorePlus to perform a shutdown in a given number of seconds. It is not necessary to perform a shutdown before the system is powered off, as it does not keep any open files while running.

**Syntax:** shutdown <seconds>

*Example*

```
Cmd> shutdown
```

## Stats

Shows various vital stats and counters.

**Syntax:** stats

*Example*

```
Cmd> stats
Uptime : 10 days, 23:11:59
Last shutdown : Unknown reason ('shutdown.txt' is empty)
```

```

CPU Load : 6%
Connections : 4919 out of 32768
Fragments : 17 out of 1024 (17 lingering)
Buffers allocated : 1252
Buffers memory : 1252 x 2292 = 2802 KB
Fragbufs allocated : 16
Fragbufs memory : 16 x 10040 = 156 KB
Out-of-buffers : 0
ARP one-shot cache : Hits : 409979144 Misses : 186865338
Interfaces: Phys:2 VLAN:5 VPN:0
Access entries:18 Rule entries:75
Using configuration file "FWCore.cfg", ver 199

```

## Sysmsg

Show the contents of the OS sysmsg buffer.

**Syntax:** sysmsg

*Example*

```

Cmd> sysmsg
Contents of OS sysmsg buffer:
2003-04-24 00:03:46 Boot device number is 0x80
2003-04-24 00:03:46 Available LowPoolMemory: 360424 Bytes
(LBlock: 360424 bytes)
2003-04-24 00:03:46 Available KernelPoolMemory: 1048560 bytes
(LBlock: 1048560 bytes)
2003-04-24 00:03:46 Available UserPoolMemory: 198868948 bytes
2003-04-24 00:03:46 Drive 0x00 present: (C/H/S/SC/M):
(0x50/0x2/0x12/0x24/0xb3f)
2003-04-24 00:03:46 Drive 0x80 present: (C/H/S/SC/M):
(0x3f2/0x10/0x33/0x330/0xc935f)
2003-04-24 00:03:46 Drive 0x80 is using a FAT-16 filesystem
2003-04-24 00:03:46 Firewall loader up and running!

```

## Time

Displays the system date and time

**Syntax:** time <options>

Options:

- set <arg> - Set system local time (YYYY-MM-DD HH:MM:SS)
- sync - Synchronize time with timeserver(s) (specified in settings)
- force - Force synchronization regardless of the MaxAdjust setting

## Uarules

Shows configured user authentication rules.

**Syntax:** uarules

*Example*

```

Cmd> uarules
Contents of the User Authentication ruleset
# Source Net Agent Auth source Authentication Server
-----
1 lan:192.168.0.0/24 HTTPAuth RADIUS FreeRadius
2 *:0.0.0.0/0 XAuth RADIUS IASRadius

```

## Userauth

**Syntax:** userauth <options>

Display information about authenticated users, known privileges.

Options:

- l - Displays a list of all authenticated users
- p - Displays a list of all known privileges (usernames and groups)
- r <ip> - Removes an authenticated user (=logout)

*Example*

```

Cmd> userauth -l
Currently authenticated users:
Login name: user1
User IP: 192.168.0.207 Idle Timeout: 1799
Privileges: members

Login name: user1
User IP: 192.168.0.214 Idle Timeout: 1800
Privileges: members

Login name: user2
User IP: 192.168.0.116 Idle Timeout: 1799
Privileges: members

```

## Userdb

**Syntax:** Userdb <dbname> [<wildcard> or <username>]

Display user databases and their contents. If <dbname> is specified users configured in that user database will be shown. A wildcard can be used to only show users matching that pattern or if a username is specified information regarding that user will be shown.

Options:

- num - Displays the specified number of users (default 20)

*Example*

```

Cmd> userdb
Configured user databases:

Name          #users
-----
LocalUsers    2

Cmd> userdb LocalUsers
Contents of user database LocalUsers:

Username  Groups  Static IP Remote Networks
-----
bob       sales
alice     tech

Cmd> userdb LocalUsers bob
Information for bob in database LocalUsers:

Username : bob
Groups   : sales
Networks :

```

## VLAN

**Syntax:** vlan

Shows information about configured VLANs.

*Example*

```
Cmd> vlan
VLANs:
vlan1 IPAddr: 192.168.123.1 ID: 1 Iface: lan
vlan2 IPAddr: 192.168.123.1 ID: 2 Iface: lan
vlan3 IPAddr: 192.168.123.1 ID: 3 Iface: lan
vlan4 IPAddr: 192.168.123.1 ID: 4 Iface: lan
```

**Syntax:** vlan <vlan>

Show information about specified VLAN.

*Example*

```
Cmd> vlan vlan1
VLAN vlan1
Iface lan, VLAN ID: 1
Iface : lan
IP Address : 192.168.123.1
Hw Address : 0003:474e:25f9
Software Statistics:
Soft received : 0 Soft sent : 0 Send failures : 0
Dropped : 0 IP Input Errs : 0
```

---

## Appendix C. fwctl command options

A summary of the command options available for the **fwctl** console management tool is shown below. An example of command option usage is:

```
> fwctl --help
```

which would display the fwctl help text. This tool is available in versions for both Linux and Windows systems.



### Note

Later versions of **fwctl** may be named **fwctl<sub>nn</sub>** where "nn" indicates the tool's version number.

## Data Source options

```
--adddsn      <dsn name> "<connect string>"
               - Add a new datasource (directory).
--rmdsn       <dsn name>
               - Remove specified datasource.
--listdsn     [--plain] [-d "<delim>"]
               - List all available datasources.
-l, --list    [--plain | -d "<delim>"] ["<pattern>"]
               - List gateways.
-d, --dumpdsn <dsn name>
               - Display contents of given DSN in tree view.
--enabledsn   <dsn name>
               - Enables datasource.
--disabledsn  <dsn name>
               - Disables datasource.
```

## Upload/Download options

```
-w, --cfgdownload [-f <file>] <gateway>
               - Download configuration file.
-u, --cfgupload   [-v <version> | -f <file>] <gateway>
               - Upload configuration file.
--coreupload     <core file> <gateway>
               - Upload (upgrade) gateway core.
--fileupload     "<source>" "<destination>" <gateway>
               - Upload any type of file.
--uploadpkg      "<source>" <gateway>
               - Uploads PKG package to gateway.
--filedownload   - Downloads specified file from gateway.
               "<source>" ["<destination>"] <gateway>
```

## Configuration management options

```
--cfginfo      [-v <version>] <gateway>
               - Display configuration information.
-o, --checkout  <gateway>
               - Mark gateway as "checked out"
-t, --printcfg  [-v <version>] <gateway>
```

```

-i, --checkin          - Display most recent configuration file.
                      [-c "<comments>"] <file> <gateway>
--undocheckout         - Check in (save) given configuration file
                      <gateway>
                      - Undo recent check out on given gateway.
--checkoutlist        [-u <user>] [-d "<delim>"]
                      - List checked-out gateways.

```

## The gateway

```

--changekeys          <gateway>
                      - Change gateway mgmt keys.
-c, --console         [-t <secs>] <gateway>
                      - Open gateway console window.
-r, --rtlog           <gateway>
                      - View real-time gateway log.
-p, --ping            [-t <secs>] <gateway>
                      - Ping specified gateway.
-e, --reconfigure     <gateway>
                      - Reconfigure specified gateway.
--restart             [--nobidir] <gateway>
                      - Restart specified gateway.
--reboot             <gateway>
                      - Hardware reboot of specified gateway.
-s, --stats           --list [-d "<delim>"] <gateway>
                      - List available statistics entries.
-s, --stats          [--count <n>] [--interval <secs>] [-d "<delim>"]
[--units]
                      [--nohdrs] [--rawcounters] <entries> <gateway>
                      - Display real-time gateway statistics.

```

## Information options

```

-v, --version        - Display version.
-h, --help           - Display command summary.

```

## Command syntax notes

```

"<gateway>" can be any of:
  <gateway name>
  <dsn name>:<gateway name>
  c:\path\to\gw-myorganization.efw

```



---

## Appendix D. IDP Signature Groups

For IDP scanning, the following signature groups are available for selection. There is a version of each group under the three *Types* of **IDS**, **IPS** and **Policy**. For further information see Section 6.3, “Intrusion Detection and Prevention”.

Group Name	Intrusion Type
APP_AMANDA	Amanda, a popular backup software
APP_ETHEREAL	Ethereal
APP_ITUNES	Apple iTunes player
APP_REALPLAYER	Media player from RealNetworks
APP_REALSERVER	RealNetworks RealServer player
APP_WINAMP	WinAMP
APP_WMP	MS Windows Media Player
AUTHENTICATION_GENERAL	Authenticantion
AUTHENTICATION_KERBEROS	Kerberos
AUTHENTICATION_XTACACS	XTACACS
BACKUP_ARKEIA	Network backup solution
BACKUP_BRIGHTSTOR	Backup solutions from CA
BACKUP_GENERAL	General backup solutions
BACKUP_NETVAULT	NetVault Backup solution
BACKUP_VERITAS	Backup solutions
BOT_GENERAL	Activities related to bots, including those controlled by IRC channels
BROWSER_FIREFOX	Mozilla Firefox
BROWSER_GENERAL	General attacks targeting web browsers/clients
BROWSER_IE	Microsoft IE
BROWSER_MOZILLA	Mozilla Browser
COMPONENT_ENCODER	Encoders, as part of an attack.
COMPONENT_INFECTIOIN	Infection, as part of an attack
COMPONENT_SHELLCODE	Shell code, as part of the attacks
DB_GENERAL	Database systems
DB_MSSQL	MS SQL Server
DB_MYSQL	MySQL DBMS
DB_ORACLE	Oracle DBMS
DB_SYBASE	Sybase server
DCOM_GENERAL	MS DCOM
DHCP_CLIENT	DHCP Client related activities
DHCP_GENERAL	DHCP protocol
DHCP_SERVER	DHCP Server related activities
DNS_EXPLOIT	DNS attacks
DNS_GENERAL	Domain Name Systems
DNS_OVERFLOW	DNS overflow attack
DNS_QUERY	Query related attacks
ECHO_GENERAL	Echo protocol and implementations
ECHO_OVERFLOW	Echo buffer overflow
FINGER_BACKDOOR	Finger backdoor
FINGER_GENERAL	Finger protocol and implementation
FINGER_OVERFLOW	Overflow for Finger protocol/implementation
FS_AFS	Andrew File System
FTP_DIRNAME	Directory name attack
FTP_FORMATSTRING	Format string attack

Group Name	Intrusion Type
FTP_GENERAL	FTP protocol and implementation
FTP_LOGIN	Login attacks
FTP_OVERFLOW	FTP buffer overflow
GAME_BOMBERCLONE	Bomberclone game
GAME_GENERAL	Generic game servers/clients
GAME_UNREAL	UnReal Game server
HTTP_APACHE	Apache httpd
HTTP_BADBLUE	Badblue web server
HTTP_CGI	HTTP CGI
HTTP_CISCO	Cisco Embedded Web Server
HTTP_GENERAL	General HTTP activities
HTTP_MICROSOFTIIS	HTTP Attacks specific to MS IIS web server
HTTP_OVERFLOWS	Buffer overflow for HTTP servers
HTTP_TOMCAT	Tomcat JSP
ICMP_GENERAL	ICMP protocol and implementation
IGMP_GENERAL	IGMP
IMAP_GENERAL	IMAP protocol/implementation
IM_AOL	AOL IM
IM_GENERAL	Instant Messenger implementations
IM_MSN	MSN Messenger
IM_YAHOO	Yahoo Messenger
IP_GENERAL	IP protocol and implementation
IP_OVERFLOW	Overflow of IP protocol/implementation
IRC_GENERAL	Internet Relay Chat
LDAP_GENERAL	General LDAP clients/servers
LDAP_OPENLDAP	Open LDAP
LICENSE_CA-LICENSE	License management for CA software
LICENSE_GENERAL	General License Manager
MALWARE_GENERAL	Malware attack
METASPLOIT_FRAME	Metasploit frame attack
METASPLOIT_GENERAL	Metasploit general attack
MISC_GENERAL	General attack
MSDTC_GENERAL	MS DTC
MSHELP_GENERAL	Microsoft Windows Help
NETWARE_GENERAL	NetWare Core Protocol
NFS_FORMAT	Format
NFS_GENERAL	NFS protocol/implementation
NNTP_GENERAL	NNTP implementation/protocol
OS_SPECIFIC-AIX	AIX specific
OS_SPECIFIC-GENERAL	OS general
OS_SPECIFIC-HPUX	HP-UX related
OS_SPECIFIC-LINUX	Linux specific
OS_SPECIFIC-SCO	SCO specific
OS_SPECIFIC-SOLARIS	Solaris specific
OS_SPECIFIC-WINDOWS	Windows specific
P2P_EMULE	eMule P2P tool
P2P_GENERAL	General P2P tools
P2P_GNUTELLA	Gnutella P2P tool
PACKINGTOOLS_GENERAL	General packing tools attack
PBX_GENERAL	PBX
POP3_DOS	Denial of Service for POP

Group Name	Intrusion Type
POP3_GENERAL	Post Office Protocol v3
POP3_LOGIN-ATTACKS	Password guessing and related login attack
POP3_OVERFLOW	POP3 server overflow
POP3_REQUEST-ERRORS	Request Error
PORTMAPPER_GENERAL	PortMapper
PRINT_GENERAL	LP printing server: LPR LPD
PRINT_OVERFLOW	Overflow of LPR/LPD protocol/implementation
REMOTEACCESS_GOTOMYPC	Goto MY PC
REMOTEACCESS_PCANYWHERE	PcAnywhere
REMOTEACCESS_RADMIN	Remote Administrator (radmin)
REMOTEACCESS_VNC-CLIENT	Attacks targeting at VNC Clients
REMOTEACCESS_VNC-SERVER	Attack targeting at VNC servers
REMOTEACCESS_WIN-TERMINAL	Windows terminal/Remote Desktop
RLOGIN_GENERAL	RLogin protocol and implementation
RLOGIN_LOGIN-ATTACK	Login attacks
ROUTER_CISCO	Cisco router attack
ROUTER_GENERAL	General router attack
ROUTING_BGP	BGP router protocol
RPC_GENERAL	RFC protocol and implementation
RPC_JAVA-RMI	Java RMI
RSYNC_GENERAL	Rsync
SCANNER_GENERAL	Generic scanners
SCANNER_NESSUS	Nessus Scanner
SECURITY_GENERAL	Anti-virus solutions
SECURITY_ISS	Internet Security Systems software
SECURITY_MCAFEE	McAfee
SECURITY_NAV	Symantec AV solution
SMB_ERROR	SMB Error
SMB_EXPLOIT	SMB Exploit
SMB_GENERAL	SMB attacks
SMB_NETBIOS	NetBIOS attacks
SMB_WORMS	SMB worms
SMTP_COMMAND-ATTACK	SMTP command attack
SMTP_DOS	Denial of Service for SMTP
SMTP_GENERAL	SMTP protocol and implementation
SMTP_OVERFLOW	SMTP Overflow
SMTP_SPAM	SPAM
SNMP_ENCODING	SNMP encoding
SNMP_GENERAL	SNMP protocol/implementation
SOCKS_GENERAL	SOCKS protocol and implementation
SSH_GENERAL	SSH protocol and implementation
SSH_LOGIN-ATTACK	Password guess and related login attacks
SSH_OPENSSSH	OpenSSH Server
SSL_GENERAL	SSL protocol and implementation
TCP_GENERAL	TCP protocol and implementation
TCP_PPTP	Point-to-Point Tunneling Protocol
TELNET_GENERAL	Telnet protocol and implementation
TELNET_OVERFLOW	Telnet buffer overflow attack
TFTP_DIR_NAME	Directory Name attack
TFTP_GENERAL	TFTP protocol and implementation
TFTP_OPERATION	Operation Attack

---

<b>Group Name</b>	<b>Intrusion Type</b>
TFTP_OVERFLOW	TFTP buffer overflow attack
TFTP_REPLY	TFTP Reply attack
TFTP_REQUEST	TFTP request attack
TROJAN_GENERAL	Trojan
UDP_GENERAL	General UDP
UDP_POPUP	Pop-up window for MS Windows
UPNP_GENERAL	UPNP
VERSION_CVS	CVS
VERSION_SVN	Subversion
VIRUS_GENERAL	Virus
VOIP_GENERAL	VoIP protocol and implementation
VOIP_SIP	SIP protocol and implementation
WEB_CF-FILE-INCLUSION	Coldfusion file inclusion
WEB_FILE-INCLUSION	File inclusion
WEB_GENERAL	Web application attacks
WEB_JSP-FILE-INCLUSION	JSP file inclusion
WEB_PACKAGES	Popular web application packages
WEB_PHP-XML-RPC	PHP XML RPC
WEB_SQL-INJECTION	SQL Injection
WEB_XSS	Cross-Site-Scripting
WINS_GENERAL	MS WINS Service
WORM_GENERAL	Worms
X_GENERAL	Generic X applications

---

## Appendix E. Anti-Virus MIME filetypes

For Anti-virus scanning, the following MIME filetypes can be checked to make sure that the content matches the filetype of a file download. Checking is done only if the option is enabled as described in Section 6.4.6, “Anti-Virus Options”.

Filetype extension	Application
3ds	3d Studio files
3gp	3GPP multimedia file
aac	MPEG-2 Advanced Audio Coding File
ab	Applix Builder
ace	ACE archive
ad3	Dec systems compressed Voice File
ag	Applix Graphic file
aiff, aif	Audio Interchange file
am	Applix SHELF Macro
arc	Archive file
alz	ALZip compressed file
avi	Audio Video Interleave file
arj	Compressed archive
ark	QuArk compressed file archive
arq	Compressed archive
as	Applix Spreadsheet file
asf	Advanced Streaming Format file
ash	Shell script
avr	Audio Visual Research Sound
aw	Applix Word file
awk	AWK Script
bash	Bash shell script
bh	Blackhole archive format file
bmp	Windows Bitmap Graphics
book	FrameMaker Book
box	VBOX voice message file
bsa	BSARC Compressed archive
bz, bz2	Bzip UNIX compressed file
cab	Microsoft Cabinet file
cdr	Corel Vector Graphic Drawing file
cgm	Computer Graphics Metafile
chz	ChArc compressed file archive
class	Java byte code
cmf	Creative Music file
core/coredump	Unix core dump
cpl	Windows Control Panel Extension file
Csh	C shell script
dbm	Database file
dcx	Graphics Multipage PCX Bitmap file
deb	Debian Linux Package file
djvu	DjVu file
dll	Windows dynamic link library file
dpa	DPA archive data
dvi	TeX Device Independent Document

Filetype extension	Application
eet	EET archive
egg	Allegro datafile
elc	eMac's Lisp Byte-compiled Source Code
emd	ABT EMD Module/Song Format file
esp	ESP archive data
exe	Windows Executable
fgf	Free Graphics Format file
flac	Free Lossless Audio Codec file
flc	FLIC Animated Picture
fli	FLIC Animation
flv	Macromedia Flash Video
gawk	GAWK script
gdbm	Database file
gif	Graphic Interchange Format file
gzip, gz, tgz	Gzip compressed archive
hap	HAP archive data
hpk	HPack compressed file archive
hqx	Macintosh BinHex 4 compressed archive
icc	Kodak Color Management System, ICC Profile
icm	Microsoft ICM Color Profile file
ico	Windows Icon file
imf	Imago Orpheus module sound data
Inf	Sidplay info file
it	Impulse Tracker Music Module
java	Java source code
jar	Java JAR archive
jng	JNG Video Format
jpg, jpeg, jpe, jff, jfif, jif	JPEG file
jrc	Jrchive compressed archive
jsw	Just System Word Processor Ichitaro
kdelnk	KDE link file
ksh	Shell script
latex	LaTeX Source Document
lha	LHA compressed archive file
lim	Limit compressed archive
lisp	LIM archive data
lzh	LZH compressed archive file
maker	FrameMaker Print file
md	MDCD compressed archive file
mdb	Microsoft Access Database
mid, midi	Musical Instrument Digital Interface MIDI-sequence Sound
mif	FrameMaker MIF file
mmf	Yamaha SMAF Synthetic Music Mobile Application Format
mml	FrameMaker MML file
mng	Multi-image Network Graphic Animation
mod	Ultratracker module sound data
mp3	MPEG Audio Stream, Layer III
mp4	MPEG-4 Video file
mpg, mpeg	MPEG 1 System Stream , Video file
mpv	MPEG-1 Video file
Microsoft files	Microsoft office files, and other microsoft files

Filetype extension	Application
msa	Atari MSA archive data
msg	Saved news text
nawk	NAWK script
niff, nif	Navy Interchange file Format Bitmap
noa	Nancy Video CODEC
nsf	NES Sound file
obj, o	Windows object file, linux object file
ocx	Object Linking and Embedding (OLE) Control Extension
ogg	Ogg Vorbis Codec compressed WAV file
out	Linux executable
pac	CrossePAC archive data
pbm	Portable Bitmap Format Image
pbf	Portable Bitmap Graphic
pdf	Acrobat Portable Document Format
pe	Portable Executable file
pfb	PostScript Type 1 Font
pgm	Portable Graymap Graphic
pgp	Pretty Good Privacy files
pkg	SysV R4 PKG Datastreams
pl	PERL Program file
pll	PAKLeo archive data
pma	PMarc archive data
png	Portable (Public) Network Graphic
ppm	PBM Portable Pixelmap Graphic
ps	PostScript file
psa	PSA archive data
psd	Photoshop Format file
qt, mov, moov	QuickTime Movie file
qxd	QuarkXpress Document
ra, ram	RealMedia Streaming Media
rar	WinRAR compressed archive
rbs	ReBirth Song file
riff, rif	Microsoft Audio file
rm	RealMedia Streaming Media
rpm	RedHat Package Manager
rtf, wri	Rich Text Format file
sar	Streamline compressed archive
sbi	SoundBlaster instrument data
sc	SC spreadsheet
sgi	Silicon Graphics IRIS Graphic file
sh	Shell script
sid	Commodore64 (C64) Music file (SID file)
sit	Stuffit archives
sky	SKY compressed archive
snd, au	Sun/NeXT audio file
so	UNIX Shared Library file
sof	ReSOF archive
sqw	SQWEZ archive data
sqz	Squeeze It archive data
stm	Scream Tracker v2 Module
svg	Scalable Vector Graphics file

---

<b>Filetype extension</b>	<b>Application</b>
svr4	SysV R4 PKG Datastreams
swf	Macromedia Flash Format file
tar	Tape archive file
tclsh	Shell script
tex	TeX Source
tfm	TeX font metric data
tiff, tif	Tagged Image Format file
tnef	Transport Neutral Encapsulation Format
torrent	BitTorrent Metainfo file
ttf	TrueType Font
txw	Yamaha TX Wave audio files
ufa	UFA archive data
vcf	Vcard file
viv	VivoActive Player Streaming Video file
wav	Waveform Audio
wk	Lotus 1-2-3 document
wmv	Windows Media file
wrl, vrml	Plain Text VRML file
xcf	GIMP Image file
xm	Fast Tracker 2 Extended Module , audio file
xml	XML file
xmcd	xmcd database file for kscd
xpm	BMC Software Patrol UNIX Icon file
yc	YAC compressed archive
zif	ZIF image
zip	Zip compressed archive file
zoo	ZOO compressed archive file
zpk	ZPack archive data
zsh	Shell script
z	Unix compressed file



---

## Appendix F. The OSI framework

The Open Systems Interconnection Model defines a framework for intercomputer communications. It categorizes different protocols for a great variety of network applications into seven smaller, more manageable layers. The model describes how data from an application in one computer can be transferred through a network medium to an application on another computer.

Control of data traffic is passed from one layer to the next, starting at the application layer in one computer, proceeding to the bottom layer, traversing over the medium to another computer and then delivering up to the top of the hierarchy. Each layer handles a certain set of protocols, so that the tasks for achieving an application can be distributed to different layers and be implemented independently.

**Figure F.1. The 7 layers of the OSI model**

Layer number	Layer purpose
Layer 7	Application
Layer 6	Presentation
Layer 5	Session
Layer 4	Transport
Layer 3	Network
Layer 2	Data-Link
Layer 1	Physical

The different layers perform the following functions:

<b>Application Layer</b>	Defines the user interface that supports applications directly. Protocols: HTTP, FTP, DNS, SMTP, Telnet, SNMP, etc.
<b>Presentation Layer</b>	Translates the various applications to uniform network formats that the rest of the layers can understand.
<b>Session Layer</b>	Establishes, maintains and terminates sessions across the network. Protocols: NetBIOS, RPC, etc.
<b>Transport Layer</b>	Controls data flow and provides error-handling. Protocols: TCP, UDP, etc.
<b>Network Layer</b>	Performs addressing and routing. Protocols: IP, OSPF, ICMP, IGMP, etc.
<b>Data-Link Layer</b>	Creates frames of data for transmission over the physical layer and includes error checking/correction. Protocols: Ethernet, PPP, etc.
<b>Physical Layer</b>	Defines the physical hardware connection.

---

# Alphabetical Index

## A

- about, CLI command, 296
- access, CLI command, 296
- access rules, 111
- accounting, 21
  - interim messages, 23
  - limitations, 24
  - messages, 21
  - system shutdowns, 24
- address book, 32
  - ethernet addresses in, 34
  - IP addresses in, 32
- address groups, 34
- address translation, 165
- AlarmRepeatInterval, 288
- AllowAuthIfNoAccountingResponse, 284
- anti-virus, 140
  - activating, 141
  - database, 141
  - memory requirements, 140
  - simultaneous scans, 140
- ARP, 48
  - proxy, 72
  - static, 49
- arp, CLI command, 296
- ARPBroadcast, 255
- ARPCacheSize, 255
- ARPChanges, 254
- ARPExpire, 255
- ARPExpireUnknown, 255
- ARPHashSize, 255
- ARPHashSizeVLAN, 255
- ARPIPCollision, 255
- ARPMatchEnetSender, 254
- ARPMulticast, 255
- ARPQueryNoSenderIP, 254
- ARPRequests, 254
- ARPSenderIP, 254
- arpsnoop, CLI command, 296
- AutoAddMulticastCoreRoute, 286
- auto-update, 29

## B

- bandwidth guarantees, 227
- blacklisting
  - hosts and networks, 163
  - IDP, 137
  - rate limiting, 232
  - URL, 145
  - wildcarding, 145
- Block0000Src, 246
- Block0Net, 246
- Block127Net, 246
- blocking applications with IDP, 134
- BlockMulticastSrc, 246
- BPDU relaying, 104
- BroadcastEnetSender, 273
- buffers, CLI command, 297

- BufFloodRebootTime, 291

## C

- certcache, CLI command, 298
- certification authority, 58
- cfglog, CLI command, 298
- CLI, 11
- cluster heartbeats, 241
- ClusterID, 276
- cluster ID, 243
- connections, CLI command, 298
- ConnLife\_IGMP, 259
- ConnLife\_Other, 259
- ConnLife\_Ping, 259
- ConnLife\_TCP, 259
- ConnLife\_TCP\_FIN, 259
- ConnLife\_TCP\_SYN, 259
- ConnLife\_UDP, 259
- ConnReplace, 257
- content filtering, 144
  - active content, 144
  - categories, 151
  - dynamic, 146
  - phishing, 155
  - spam, 157
  - static, 145
- core interface, 41
- cpuid, CLI command, 299

## D

- date and time setting, 60
- DefaultTTL, 247
- denial of service, 158
- DHCP, 107
  - over ethernet, 43
  - relaying, 109
  - servers, 108
- DHCP\_AllowGlobalBcast, 269
- DHCP\_DisableArpOnOffer, 269
- DHCP\_MinimumLeaseTime, 269
- DHCP\_UseLinkLocalIP, 269
- DHCP\_ValidateBcast, 269
- dhcp, CLI command, 299
- DHCPRelay\_AutoSaveRelayInterval, 270
- DHCPRelay\_MaxAutoRoutes, 270
- DHCPRelay\_MaxHops, 270
- DHCPRelay\_MaxLeaseTime, 270
- DHCPRelay\_MaxPPMPerIface, 270
- DHCPRelay\_MaxTransactions, 270
- DHCPRelay\_TransactionTimeout, 270
- dhcrelay, CLI command, 299
- DHCPServer\_AutoSaveLeaseInterval, 271
- DHCPServer\_SaveLeasePolicy, 271
- DHCPServer\_SaveRelayPolicy, 270
- dhcpserver, CLI command, 300
- DirectedBroadcasts, 247
- distribution algorithms, 234
- DNS, 63
  - DNS\_DNSServerIP1, 279
  - DNS\_DNSServerIP2, 279
  - DNS\_DNSServerIP3, 279
- DroppedFragments, 263

DuplicateFragData, 262  
 DuplicateFragments, 263  
 dynamic bandwidth balancing, 231  
 dynamic routing policy, 88  
 dynroute, CLI command, 300

**E**

ethernet, 41  
   IP addresses, 42  
 events, 16  
   distribution, 16  
   messages, 16

**F**

FragmentedICMP, 264  
 FragReassemblyFail, 263  
 frags, CLI command, 300

**G**

GRE, 46  
   uses, 46

**H**

ha, CLI command, 301  
 HAINitialSilence, 276  
 HASyncBufSize, 276  
 HASyncMaxPktBurst, 276  
 high availability, 239  
 HighBuffers, 291  
 HTTPPoster\_RepDelay, 281  
 HTTPPoster\_URL1, HTTPPoster\_URL2, HTTP-  
 Poster\_URL3, 281  
 httpposter, CLI command, 301  
 HWM\_PollInterval, 288  
 HWMMem\_AlertLevel, 288  
 HWMMem\_CriticalLevel, 288  
 HWMMem\_Interval, 288  
 HWMMem\_LogRepetition, 288  
 HWMMem\_UsePercent, 288  
 HWMMem\_WarningLevel, 288

**I**

ICMPSendPerSecLimit, 253  
 icons, xii  
 IDP (see intrusion, detection and prevention)  
 IDP\_UpdateInterval, 285  
 ifacegroups, CLI command, 301  
 ifstat, CLI command, 302  
 IGMPBeforeRules, 286  
 IGMPLastMemberQueryInterval, 287  
 IGMPLowestCompatibleVersion, 286  
 IGMPMaxReqs, 286  
 IGMPMaxReqsIf, 286  
 IGMPQueryInterval, 286  
 IGMPQueryResponseInterval, 287  
 IGMPReactToOwnQueries, 286  
 IGMPRobustnessVariable, 286  
 IGMPRouterVersion, 286  
 IGMPStartupQueryCount, 287  
 IGMPStartupQueryInterval, 287

IGMPUnsolicitedReportInterval, 287  
 IKE, 186  
 IKECRLValidityTime, 272  
 IKEMaxCAPath, 272  
 IKESendCRLs, 272  
 IKESendInitialContact, 272  
 ikesnoop  
   troubleshooting with, 206  
 ikesnoop, CLI command, 303  
 IllegalFragments, 262  
 interfaces, 40  
   groups, 47  
 intrusion, detection and prevention, 134  
   signature groups, 136  
 intrusion detection rule, 134  
 IPOPT\_OTHER, 247  
 IPOPT\_SR, 247  
 IPOPT\_TS, 247  
 IPOptionSizes, 247  
 IPRF, 247  
 IP rules  
   actions, 54  
   evaluation order, 53  
 ip rule-set, 53  
 IPsec, 186  
 IPsecBeforeRules, 272  
 IPsecCertCacheMaxCerts, 272  
 ipseckeepalive, CLI command, 303  
 ipsecstats, CLI command, 303  
 ipsectunnels, CLI command, 303

**L**

L2TP, 215  
 Lan to Lan tunnels, 199  
 LayerSizeConsistency, 247  
 license, CLI command, 304  
 link state algorithms, 84  
 LocalReass\_MaxConcurrent, 266  
 LocalReass\_MaxSize, 266  
 LocalReass\_NumLarge, 266  
 lockdown, CLI command, 304  
 LogChecksumErrors, 246  
 LogConnections, 257  
 loghosts, CLI command, 305  
 LogNonIP4, 246  
 LogOpenFails, 257  
 logout, CLI command, 305  
 LogOutAccUsersAtShutdown, 284  
 LogOversizedPackets, 261  
 LogReceivedTTL0, 246  
 LogReverseOpens, 257  
 LogSendPerSecLimit, 275  
 LogStateViolations, 257  
 loopback, , 46

**M**

MAC address, 48  
 MaxAHLen, 260  
 MaxConnections, 257  
 MaxESPLen, 260  
 MaxGRELen, 260  
 MaxICMPLen, 260

MaxIPCompLen, 261  
 MaxIPIPLen, 261  
 MaxL2TPLen, 261  
 MaxOSPFLen, 261  
 MaxOtherSubIPLen, 261  
 MaxPipeUsers, 291  
 MaxSKIPLen, 261  
 MaxTCPLen, 260  
 MaxUDPLen, 260  
 memory, CLI command, 305  
 MinimumFragLength, 264  
 MulticastEnetSender, 274  
 multicast on ethernet interfaces, 42  
 multicast routing, 91

## N

NAT, 165  
 netcon, CLI command, 305  
 NetconBeforeRules, 280  
 NetConBiDirTimeout, 280  
 netobjects, CLI command, 305  
 NullEnetSender, 273

## O

OSPF, 85  
   aggregates, 86  
   table, 48  
 ospf, CLI command, 305  
 overriding content filtering, 149

## P

packet flow, 3  
   diagram, 6  
 phishing (see content filtering)  
 ping, CLI command, 306  
 pipes, 222  
 pipes, CLI command, 306  
 policies, 53  
 policy based routing, 73  
 PPOE, 44  
   client configuration, 45  
 PPP\_L2TPBeforeRules, 283  
 PPP\_PPTPBeforeRules, 283  
 PPTP, 214  
 proplists, CLI command, 307  
 PseudoReass\_MaxConcurrent, 262

## R

RADIUS  
   accounting, 21  
   authentication, 180  
 rapid failover, 241  
 rate limiting, 232  
 ReassDoneLinger, 264  
 Reassembly\_MaxConnections, 290  
 Reassembly\_MaxProcessingMem, 290  
 ReassIllegalLinger, 264  
 ReassTimeLimit, 264  
 ReassTimeout, 264  
 reconfigure, CLI command, 307  
 RelaySTP, 274

remotes, CLI command, 308  
 reset to factory defaults, 29  
 RFO\_GratuitousARPOnFail, 276  
 Ringsize\_e100\_rx, 282  
 Ringsize\_e100\_tx, 282  
 Ringsize\_e1000\_rx, 282  
 Ringsize\_e1000\_tx, 282  
 route failover, 70  
 route notation, 67  
 routes, CLI command, 308  
 routing, 65  
   dynamic, 84  
   metrics, 84  
   monitoring, 70  
   static, 66  
 rules, CLI command, 309

## S

safestream, 141  
 SAT, 168  
 schedules, 56  
 sersave, CLI command, 309  
 ScrSaveTime, 291  
 server load balancing, 233  
 services, 36  
   custom, 39  
   ICMP, 38  
 services, CLI command, 309  
 settings, CLI command, 309  
 shared IP address in HA, 241  
 shutdown, CLI command, 310  
 SilentlyDropStateICMPErrors, 253  
 SNMP, 18  
   traps, 18  
 SNMPBeforeRules, 280  
 SNMPifAlias, 268  
 SNMPifDescr, 268  
 SNMPReqLimit, 268  
 SNMPSysContact, 268  
 SNMPSysLocation, 268  
 SNMPSysName, 268  
 spam (see content filtering)  
 spanning tree relaying, 104  
 spoofing, 111  
 StaticARPChanges, 254  
 stats, CLI command, 310  
 StripDFOnSmall, 248  
 synchronization interface, 242  
 syslog, 17  
 sysmsgs, CLI command, 311

## T

TCPECN, 251  
 TCPFinUrg, 251  
 TCPMSSAutoClamping, 249  
 TCPMSSLogLevel, 249  
 TCPMSSMax, 249  
 TCPMSSMin, 249  
 TCPMSSOnHigh, 249  
 TCPMSSOnLow, 249  
 TCPMSSVPNMax, 249  
 TCPNULL, 252

TCPOPT\_ALTCHKDATA, 250  
 TCPOPT\_ALTCHKREQ, 250  
 TCPOPT\_CC, 251  
 TCPOPT\_OTHER, 251  
 TCPOPT\_SACK, 250  
 TCPOPT\_TSOPT, 250  
 TCPOPT\_WSOPT, 250  
 TCPOptionSizes, 249  
 TCPRF, 252  
 TCPSynPsh, 251  
 TCPSynUrg, 251  
 TCPUrg, 251  
 TCPZeroUnusedACK, 250  
 TCPZeroUnusedURG, 250  
 time, CLI command, 311  
 TimeSync\_DSTEnabled, 277  
 TimeSync\_DSTEndDate, 278  
 TimeSync\_DSTOffs, 277  
 TimeSync\_DSTStartDate, 278  
 TimeSync\_GroupIntervalSize, 277  
 TimeSync\_MaxAdjust, 277  
 TimeSync\_ServerType, 277  
 TimeSync\_SyncInterval, 277  
 TimeSync\_TimeServerIP1, 277  
 TimeSync\_TimeServerIP2, 277  
 TimeSync\_TimeServerIP3, 277  
 TimeSync\_TimeZoneOffs, 277  
 time synchronization, 61  
 traffic shaping, 220  
 Transp\_CAMToL3CDESTLearning, 273  
 Transp\_DecrementTTL, 273  
 Transparency\_ATSExpire, 274  
 Transparency\_ATSSize, 274  
 Transparency\_CAMSize, 273  
 Transparency\_L3CSize, 273  
 transparent mode, 99  
 TTLMin, 246  
 TTLOnLow, 246  
 tunnels, 40

## U

uarules, CLI command, 311  
 UnknownVLANTags, 267  
 UnsolicitedARPreplies, 254  
 userauth, CLI command, 311  
 userdb, CLI command, 312

## W

whitelisting
 

- hosts and networks, 163
- URL, 145
- wildcarding, 145

 wildcarding
 

- in blacklists and whitelists, 145
- in IDP rules, 136

## V

virtual LAN, 43  
 virtual links, 86  
 virus scanning, 140  
 VPNs, 184

## X

X.509 certificates, 58