Administrators Guide

Clavister® CorePlus™
Version 9.00.03

Clavister AB
Sjögatan 6J
SE-89160 Örnsköldsvik
SWEDEN

Phone: +46-660-299200
Fax: +46-660-12250

www.clavister.com

Build: 90003
Published 2009-02-26

Administrators Guide
Clavister® CorePlus™
Version 9.00.03

Published 2009-02-26
Build: 90003

Copyright Notice

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

List of Figures

List of Examples

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 sub-sections. Numbered sub-sections 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 chapter/section" link (such as: see Chapter 10, *VPN*) 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 a italics*.

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

```
Device:/>
```

Where a web address reference is shown in the text, clicking it will open the specified URL in a browser in a new window (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 CLI example and/or a Web Interface example as appropriate. (The CorePlus *CLI Reference Guide* documents all CLI commands.)

Example 1. Example Notation

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

CLI
The Command Line Interface example would appear here. It would start with the command prompt followed by the command:

```
Device:/> somecommand someparameter=somevalue
```

Web Interface

The Web Interface actions for the example are shown here. They are also typically a numbered list showing what items in the tree-view list at the left of the interface or in the menu bar or in a context menu need to be opened followed by information about the data items that need to be entered:

1. Go to **Item X > Item Y > Item Z**
2. Now enter:
   - **DataItem1:** datavalue1
   - **DataItem2:** datavalue2

Feedback

Despite all efforts, unintentional typographic errors, factual errors or omissions may regretably occur in documentation. Clavister appreciates all feedback pointing out such problems or any other suggestions for improvement of the content. Please email all feedback to *mailto:support@clavister.com*.

Highlighted Content

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 are of the following types with the following purposes:

Note
This indicates some piece of information that is an addition to the preceding text. It may concern something that is being emphasized, 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. CorePlus Overview

This chapter outlines the key features of CorePlus.

1.1. Features

Clavister CorePlus is the firmware, the software engine that drives and controls all Clavister Security Gateway products.

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, , virtual routing as well as multicast routing capabilities. In addition, CorePlus supports features such as Virtual LANs, Route Monitoring, Proxy ARP and Transparency. For more information, please see Chapter 5, *Routing*.

Firewalling Policies — CorePlus provides stateful inspection-based firewalling for a wide range of protocols such as TCP, UDP and ICMP. The administrator can define detailed firewalling policies based on source/destination network/interface, protocol, ports, user credentials, time-of-day and more. Section 4.5, “The IP Rule Set”, describes how to set up these policies to determine what traffic is allowed or rejected by CorePlus.

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 8, *Address Translation*.

VPN — CorePlus supports a range of Virtual Private Network (VPN) solutions. 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. The details for this can be found in Chapter 10, *VPN* which includes a summary of setup steps in Section 10.2, “VPN Quick Start”.

ALGs

CorePlus provides a range of Application Level Gateways (ALGs) which provide security features that examine traffic at higher OSI layers such checking if file download content agrees with the given filetype. Another example is the SIP ALG that is able to handle the SIP message exchanges that take place during the setup of peer to peer data exchanges. For detailed information, see Section 7.2, “ALGs”.

Anti-Virus Scanning

CorePlus features integrated anti-virus functionality. Traffic passing through the Clavister Security Gateway can be subjected to in-depth scanning for viruses, and virus sending hosts can be black-listed and blocked. For details of this feature, see Section 7.4, “Anti-Virus Scanning”.

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 information about the IDP capabilities of CorePlus, please see Section 7.5, “Intrusion Detection and Prevention”.

Web Content Filtering

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 on filtering, see Section 7.3, “Web Content Filtering”.

Traffic Management

CorePlus provides broad traffic management capabilites through Traffic Shaping, Threshold Rules and Server Load Balancing. Traffic Shaping enables limiting and balancing of bandwidth; Threshold Rules allow specification of thresholds for sending alarms and/or limiting network traffic; Server Load Balancing enables a device running CorePlus to distribute network load to multiple hosts. These features are discussed in detail in Chapter 11, *Traffic Management*.

User Authentication

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 9, *User Authentication*, provides detailed information about this feature.

Operations and Maintenance

Adminstrator management of CorePlus is possible through either a Web-based User Interface (the WebUI) or via a Command Line Interface (the CLI). CorePlus also provides detailed event and logging capabilities plus support for monitoring through SNMP. For more detailed information, see Chapter 3, *Management and Maintenance*.

High Availability

High Availability (HA) is supported through automatic fault-tolerant fail-over to a secondary Clavister Security Gateway device. The two devices act together as a *cluster*, with one being active while the other is passive but constantly mirroring the state of the active unit. This feature is described in more detail in Chapter 12, *High Availability*.

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.

CorePlus Documentation

Reading through the available 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 reference guides:

- The <emphasis>CLI Reference Guide</emphasis> which details all CorePlus CLI commands.
- The *CorePlus Log Reference Guide* which details all CorePlus log event messages.

Together, these documents form the essential reference material for CorePlus operation.

CorePlus Education and Certification

For details about classroom and online CorePlus education as well as CorePlus certification, visit the Clavister company website at *http://www.clavister.com* or contact your local sales representative.

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 eliminates any possibility to detect and analyze complex protocols and enforce corresponding security policies.

Stateful Inspection

CorePlus employs a technique called *stateful inspection* which means that it inspects and forwards traffic on a per-connection basis. CorePlus detects when a new connection is being established, and keeps a small piece of information or *state* in its *state table* for the lifetime of that connection. By doing this, CorePlus is able to understand the context of the network traffic, which enables it to perform in-depth traffic scanning, apply bandwidth management and much more.

The stateful inspection approach additionally provides high throughput performance with the added advantage of a design that is highly scalable. The CorePlus subsystem that implements stateful inspection will sometimes be referred to in documentation as the CorePlus *state-engine*.

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

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.

Interface Symmetry

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

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 which represent specific protocol and port combinations. Also important are the Application Layer Gateway (ALG) objects which are used to define additional parameters on specific protocols such as HTTP, FTP, SMTP and H.323.

CorePlus Rule Sets

Finally, rules which are defined by the administrator in the various *rule sets* are used for actually implementing CorePlus security policies. The most fundamental set of rules are the *IP Rules*, which are 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 IDP Rules control the behavior of the intrusion prevention engine and so on.

1.2.3. Basic Packet Flow

This section outlines the basic flow in the state-engine for packets received and forwarded by CorePlus. The following description is simplified and might not be fully applicable in all scenarios, however, the basic principles will be valid for all CorePlus deployments.

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 on. 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 and IP protocol.

   If a match cannot be found, a connection establishment process starts which includes steps from here 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. Policy-based Routing Rules are also 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. The following parameters are part of the matching process:

   - Source and destination interfaces

   - Source and destination network

   - IP protocol (for example TCP, UDP, ICMP)

   - TCP/UDP ports

   - ICMP types

- Point in time in reference to a pre-defined schedule

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 Threshold Limit 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 on, to further analyze or transform the traffic.

    - If the contents of the packet is encapsulated (such as with IPsec, PPTP/L2TP or some other type of tunneled protocol), then 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.

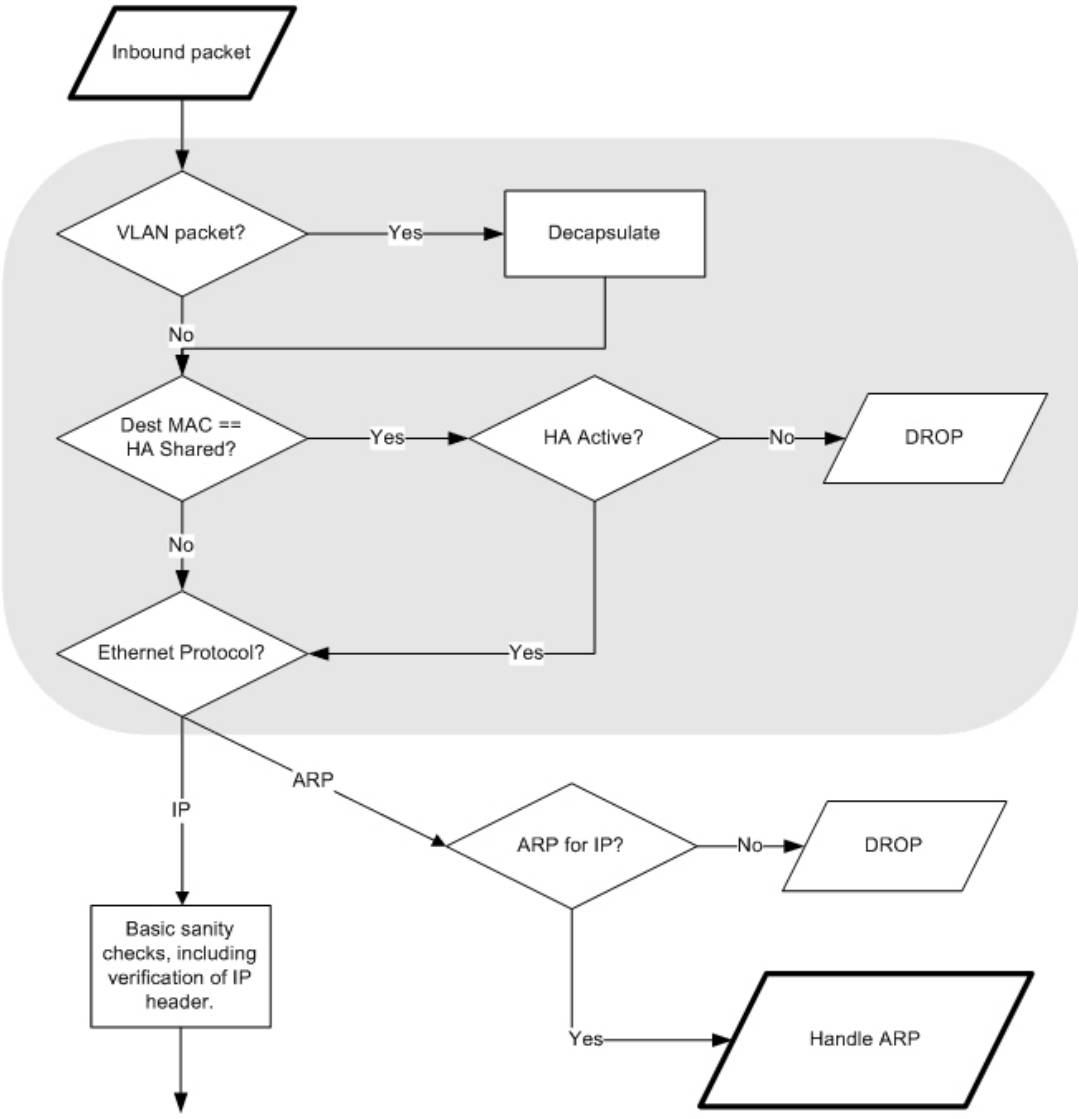
12. Eventually, the packet will be forwarded out on the destination interface according to the state. If the destination interface is a tunnel interface or a physical sub-interface, additional processing such as encryption or encapsulation might occur.

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

1.3. CorePlus State Engine Packet Flow

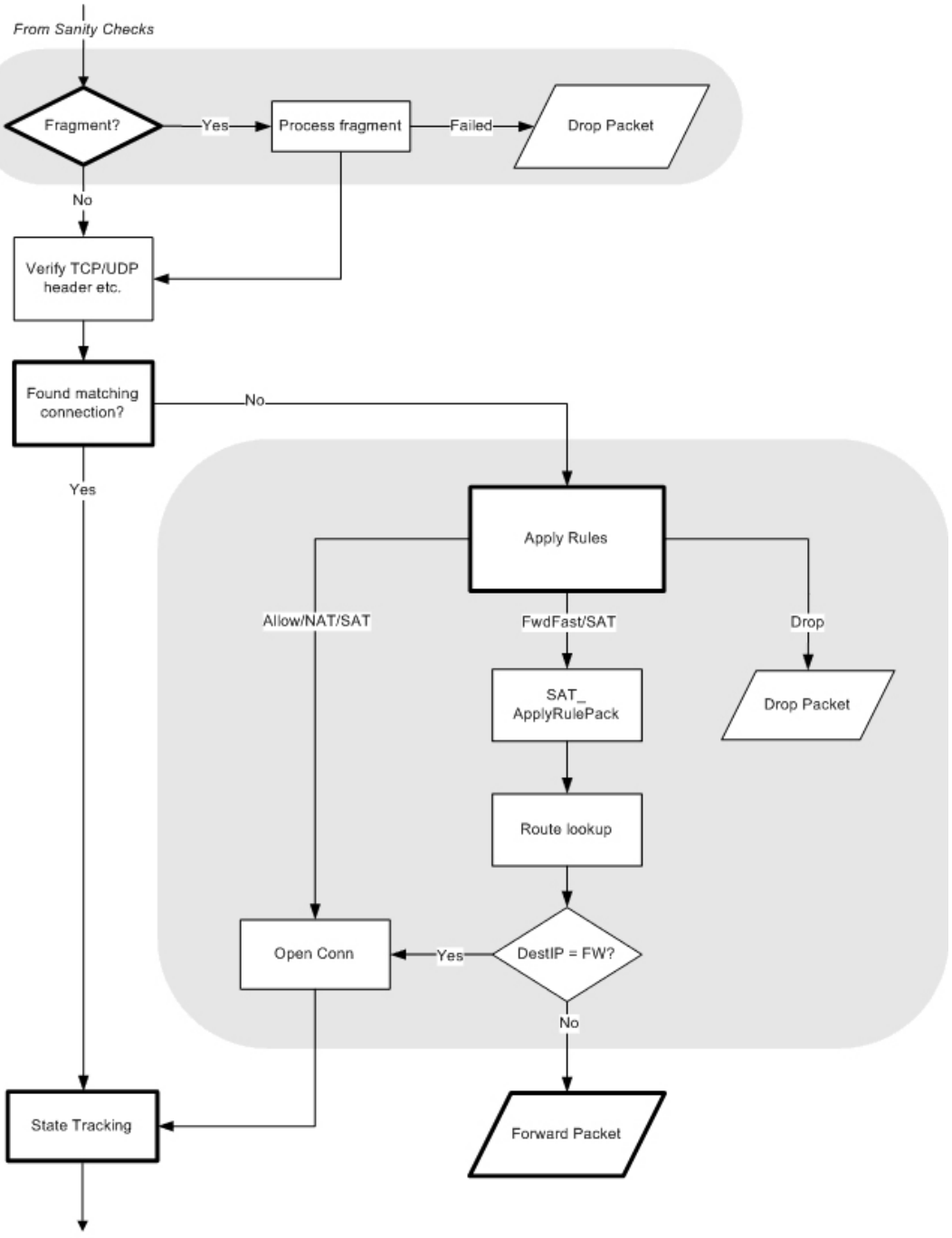
The diagrams in this section provide a summary of the flow of packets through the CorePlus state-engine. There are three diagrams, each flowing into the next.

Figure 1.1. Packet Flow Schematic Part I



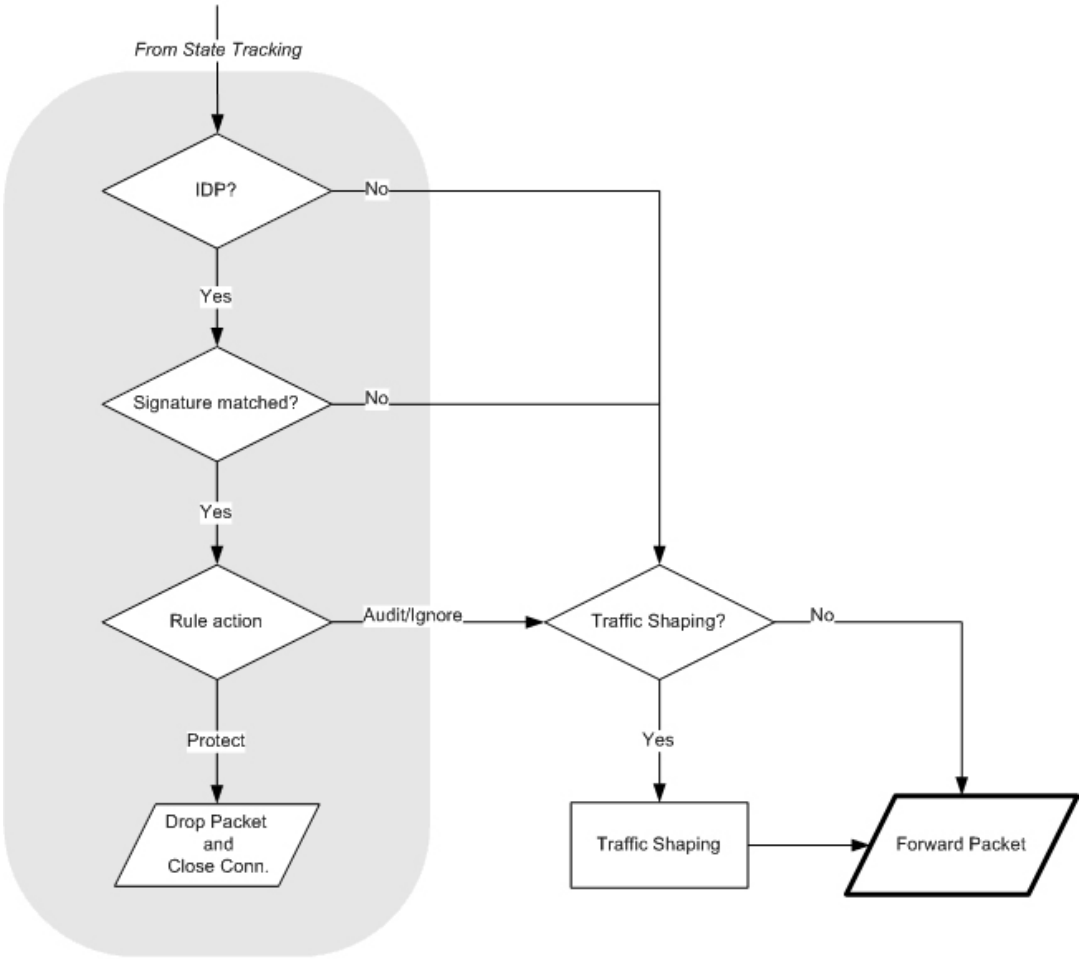
The packet flow is continued on the following page.

Figure 1.2. Packet Flow Schematic Part II



The packet flow is continued on the following page.

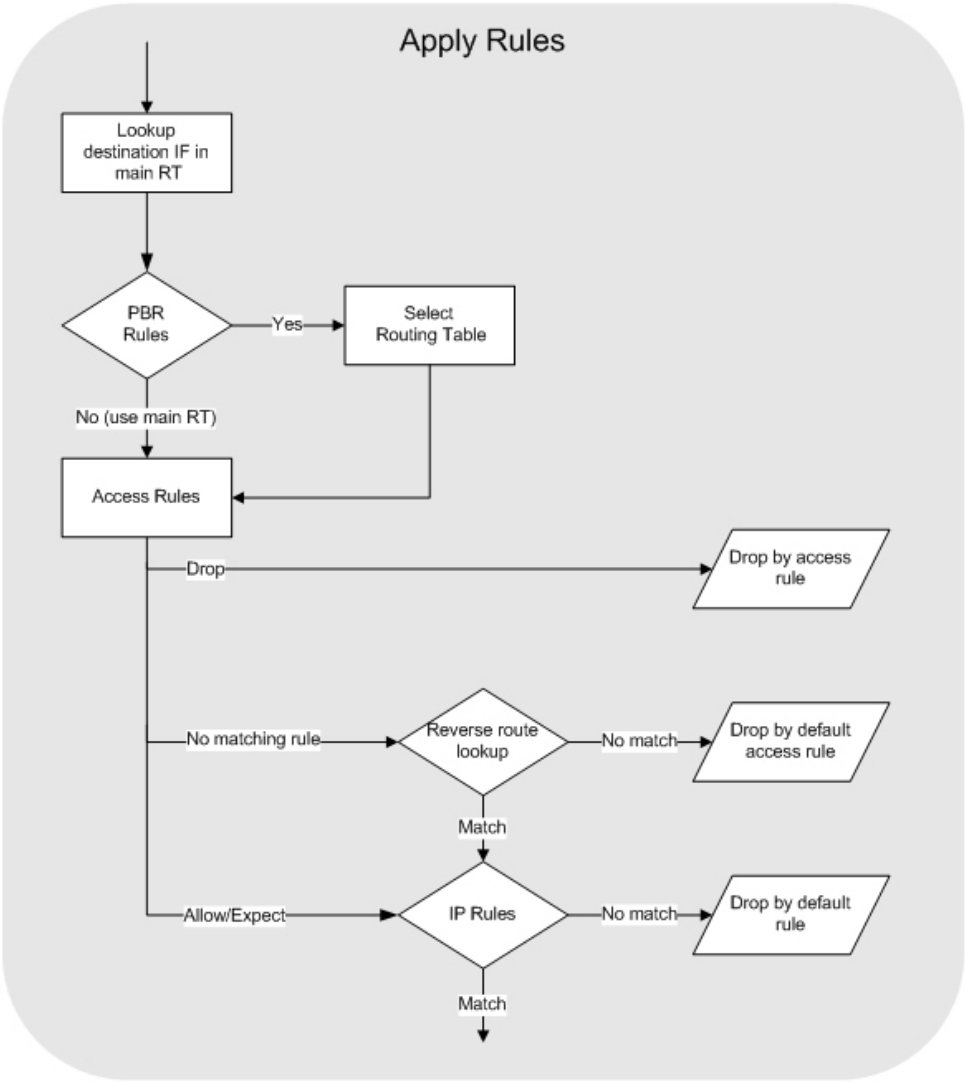
Figure 1.3. Packet Flow Schematic Part III



Apply Rules

The figure below presents the detailed logic of the *Apply Rules* function in Figure 1.2, “Packet Flow Schematic Part II” above.

Figure 1.4. Expanded *Apply Rules* Logic



Chapter 2. Upgrading to 9.nn

This section deals with the procedure for upgrading an older 8.nn version of CorePlus to a newer 9.nn CorePlus version. All CorePlus 8.nn versions from and including 8.60.01 upwards can be upgraded to a 9.nn version. This section should be skipped if any CorePlus 9.nn version is already installed.

Running *upgrade.exe*

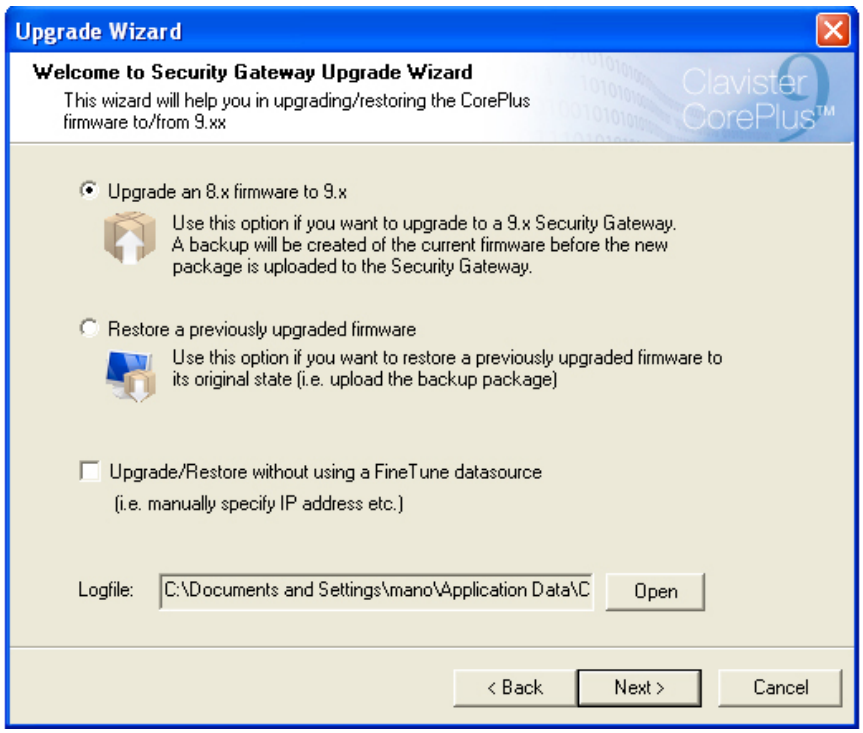
In all 9.nn distributions a separate Microsoft Windows PC executable file is included called *upgrade.exe*. This is found in the *Upgrade/Wizard* folder. Running this program will start the upgrade wizard which will take the user through the process of upgrading Clavister Security Gateways running an 8.nn version of CorePlus.

This program should be run on the same PC which is running the copy of the Clavister FineTune™ software being used for administration of the 8.nn version. This is helpful because *upgrade.exe* can then find and use the FineTune *Data Source* files which contain definition data for the Clavister Security Gateway being upgraded.

Alternatively the wizard may be run on a PC which is not also running Clavister FineTune. This is only possible with Clavister Security Gateways that are brand new or an older unit that has been reset to factory defaults. In either case no NetCon keys will be set and therefore the data source is not needed. This option is described fully below, at the very end of this section.

The Clavister Security Gateway being upgraded should be up and running but not being used to transfer live traffic since the upgrade process will interrupt and temporarily suspend CorePlus operation.

Figure 2.1. The Upgrade Wizard



Upgrade Wizard Step Sequence

The upgrade uses a familiar layout with successive dialogs controlled by forward, back and cancel buttons. The first two screens are:

- The initial start screen informs the user of the wizard's purpose.
- The next screen (shown above) offers the user a choice of:
  - Upgrade from 8.nn to 9.nn.
  - Restore from a backup.

  The first option is the default and is the one being considered in the following steps. The second option provides a way of reversing the upgrade and is discussed further on in this section.

  This screen also offers a further option to upgrade/restore without Clavister FineTune installed and that is discussed at the very end of this section.

The administrator now enters the upgrade procedure and each step and its screen is numbered:

1. **Select the unit to upgrade**
   The wizard scans the local PC to find the 8.nn data source files and lists the data sources it finds with a list of Clavister Security Gateways defined for each data source. The unit that is to be upgraded should be selected before pressing the **Next** button.

2. **Initial Processing**
   The wizard now performs the initial processing, gathering the data required for the upgrade. A unique backup file associated with the Clavister Security Gateway is created in this step if reversing the upgrade is required later. When complete press the **Next** button.

3. **Specify WebUI and/or SSH access**
   This step allows the administrator to specify the user interfaces that will be allowed, WebUI and/or SSH (Secure Shell). At least one option should be checked and the appropriate parameters should be added. The *Network* is where management traffic will come from and the *Interface* is where that traffic will arrive on the Clavister Security Gateway.

4. **Enter the administrators username and password**
   A username/password combination must be added for initial administrator access. After entering these and confirming the password, press the **Next** button.

5. **Advanced Settings and Memlog**

   Two options are now displayed:

   - *Use Default advanced settings* - If this option is checked then the default of all CorePlus advanced settings will be used in the upgraded configuration. If the option is not checked then the advanced settings will remain as they were in the old CorePlus configuration.
   - *Use Memlog* - MemLog is the facility to save a stack of log messages in memory. These can be then be directly queried through the management interface instead of needing to query a separate log server. Memlog consumes a small amount of memory which will not affect performance.

   After selection, press the **Next** button.

6. **Create Files for Uploading**
   All the files required for the upgrade are now created, validated and saved ready for uploading. After processing is complete, press the **Next** button.

7. **Select Upgrade Packet** The Clavister upgrade packet is a file which contains the executable CorePlus software version to be uploaded (sometimes referred to as the *Core*). There are different packets for different processor types and the packet file name indicates what it is intended for.

The wizard will automatically pick out the right upgrade packet to suit the hardware type from the ones it comes included with and pressing **Next** is all that is needed to go to the next step.

If a new upgrade packet has been downloaded from the Clavister Customer Web then pressing the **Browse** button will allow you to pick that packet instead of the default.

8. **Upload to complete the upgrade**
   The upgrade files are now uploaded to the Clavister Security Gateway and the unit is then rebooted. After completion, press the **Next** button.

9. **Finish screen**
   Press the **Finish** button to close the wizard. The option is given on the final screen to automatically open the WebUI in a browser window.

Reversing an Upgrade with a Wizard Created Backup

As mentioned above, an 8.nn to 9.nn upgrade automatically creates a backup file of the old 8.nn configuration. If required the upgrade wizard can then be used to undo the upgrade and restore the CorePlus back to the old 8.nn version. The backup file contains all the data needed to do this.

If the option to restore from a backup is chosen in the wizard, a single screen is displayed which shows a list of Clavister Security Gateways that have associated backup files. A backup file is uniquely associated with the Clavister Security Gateway, so it should not be necessary to change the backup file name selection. Choosing the hardware unit will be enough.

Once the Clavister Security Gateway is selected, pressing the **Next button** will cause the upgrade to be reversed, followed by an automatic reboot.

Reversing an Upgrade Manually

It should usually not be necessary to undo an upgrade manually since the wizard provides an automatic feature to do this but in unusual circumstances this might be required. The manual steps are:

1. Perform a reset to factory defaults on the hardware.

2. Open Clavister FineTune and re-establish communication with the Clavister Security Gateway.

3. Reinstall the CorePlus loader by choosing the Clavister FineTune option:
   Action > Communication > Upgrade > Loader

4. Reinstall the CorePlus core by choosing the option:
   Action > Communication > Upgrade > Core

5. Reinstall the last pre-upgrade configuration by choosing the option:
   Action > Communication > Upload > Configuration

Note on losing changes made since upgrading.
Any configuration changes made since upgrading using the wizard will be lost when reversing the process. This is because the Clavister FineTune data source for the Clavister Security Gateway will not be recording any configuration changes made since the upgrade.

Upgrading Without FineTune Installed

As explained at the beginning of this section, it is possible to upgrade a 8.nn Clavister Security

Gateway unit by running the upgrade wizard on a PC that does not also have FineTune installed. This is possible only if:

- The unit is brand new and has not yet been configured.
- The unit is not brand new but has had the factory defaults restored.

In either case, the steps to upgrade are as follows:

1. Configure the 8.nn system normally so that traffic from the PC running the upgrade wizard is accepted (in other words the upgrade wizard will communicate with CorePlus as though it were Clavister FineTune, using NetCon).
2. Below the initial upgrade wizard option to choose an upgrade or restore, tick the checkbox to indicate that **Upgrade/Restore without using any Clavister FineTune data source**.
3. An extra wizard dialog will then appear asking for the following information:
   - **Name**
     This is the name of the Clavister Security Gateway as it will appear to the administrator by all management interfaces.
   - **IP Address**
     This is the IP address of the Clavister Security Gateway that will be used for access by management interfaces.
   - **Type**
     This allows the unit to be designated as a master or slave in a high availability cluster.
   - **Backup Path**
     This is the location of the backup created during the upgrade. This path must be specified since the wizard may be running directly from non-writable media such as a CD-ROM. If the directory specified is *<path>* then the backup files will be written in: *<path>/backup/<gw-name>* where *<gw-name>* is the name specified above for the Clavister Security Gateway.
4. The wizard will now continue on as described in the first part of this section.

Chapter 3. Management and Maintenance

This chapter describes the management, operations and maintenance related aspects of CorePlus.

3.1. Managing CorePlus

3.1.1. Overview

CorePlus is designed to 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 the configuration subsystem as well as a description of how to work with the various management interfaces.

Management Interfaces

CorePlus provides the following management interfaces:

The Web Interface — The *Web Interface* (also known as the *Web User Interface* or *WebUI*) is built into CorePlus and provides a user-friendly and intuitive graphical management interface, accessible from a standard web browser (Microsoft Internet Explorer or Firefox is recommended). The browser connects to one of the hardware's Ethernet interfaces using *HTTP* or *HTTPS* and the CorePlus responds like a web server, allowing web pages to be used as the management interface.

The WebUI does not provide centralised management control of multiple Clavister Security Gateways. One browser window can communicate with one Clavister Security Gateway, although it is possible to have multiple browser windows open at the same time.

This feature is fully described in Section 3.1.3, “The Web Interface”.

The CLI — The *Command Line Interface* (CLI), accessible locally via serial console port or remotely using the Secure Shell (SSH) protocol, provides the most fine-grained control over all parameters in CorePlus.

This feature is fully described in Section 3.1.4, “The CLI”.

Secure Copy — *Secure Copy* (SCP) is a widely used communication protocol for file transfer. No specific SCP client is provided with CorePlus distributions but there exists a wide selection of SCP clients available for nearly all

workstation platforms. SCP is a complement to CLI usage and provides a secure means of file transfer between the administrator's workstation and the Clavister Security Gateway. Various files used by CorePlus can be both uploaded and downloaded with SCP.

This feature is fully described in Section 3.1.6, “Secure Copy”.

Console Boot Menu

Before CorePlus starts running, a console connected directly to the Clavister Security Gateway's RS232 port can be used to do basic configuration through the *boot menu*. This menu can be entered by pressing any console key between power-up and CorePlus starting. It is the *Clavister firmware loader* that is being accessed with the boot menu.

The menu is fully described in Section 3.1.7, “The Console Boot Menu”.

Note

Microsoft Internet Explorer (version 7 and later), Firefox (version 3.0 and later) and Netscape (version 8 and later) are the recommended web-browsers to use with the WebUI. Other browsers may also provide full support.

Remote Management Policies

Access to remote management interfaces can be regulated by a *remote management policy* so the administrator can restrict management access based on source network, source interface and username/password credentials.

3.1.2. Default Administrator Accounts

By default, CorePlus has a local user database, *AdminUsers*, which contains two pre-defined user accounts:

- Username *admin* with password *admin*.
  This account has full administrative read/write privileges.
- Username *audit* with password *audit*.
  This account is for monitoring purposes only and has read-only privileges.

Important

For security reasons, it is recommended to change the default passwords of the default accounts as soon as possible after connecting with the Clavister Security Gateway.

Creating Additional Accounts

Extra user accounts can be created as required. Accounts can either belong to the **Administrator** user group, in which case they have complete read/write administrative access. Alternatively, they can belong to the **Auditor** user group, in which case they have read-only access.

3.1.3. The Web Interface

CorePlus provides an intuitive *Web Interface* (WebUI) for management of the system via an Ethernet interface using a standard web browser. This allows the administrator to perform remote management from anywhere on a private network or the public Internet using a standard computer without having to install client software.

Assignment of a Default IP Address

For new Clavister hardware models with factory defaults, a default internal IP address of *192.168.1.1* is assigned by CorePlus to the interface indicated in the list below.

Clavister Hardware Model Series	Default WebUI Management Interface
SG10/50	lan
SG3100	if1
SG3200/4200/4300/4400	ge1
SG5500	cmm
ATCA SG6010	ge1

For non-Clavister hardware, CorePlus scans the available interfaces and allocates the *192.168.1.1* IP address to the first interface it finds. Simply trying to connect to each interface in turn with a web browser can reveal which has been selected.

Setting the Workstation IP

The assigned Clavister Security Gateway interface and the workstation interface must be on the same IP network for inital communication between them to succeed so the static IP address of the workstation must be set to the following values:

- **IP address:** *192.168.1.30*
- **Subnet mask:** *255.255.255.0*
- **Default gateway:** *192.168.1.1*

For details on how to set the static IP address for Windows and MacOS workstations, see the *CorePlus Setup Guide*.

Logging on to the Web Interface

To access the Web Interface using the factory default settings, launch a web browser on the workstation (the latest version of Internet Explorer or Firefox is recommended) and point the browser at the address *192.168.1.1*.

When performing initial connection to CorePlus, the administrator should use **https://** as the URL protocol in the browser (in other words, **https://192.168.1.1**). Using HTTPS as the protocol makes communication with CorePlus secure.

The *http://* protocol can be used instead, for logon but this is not secure and should be used only across internal networks.

If communication with the CorePlus is successfully established, a user authentication dialog similar to the one shown below will then be shown in the browser window.

After entering a valid username and password the **Login** button is clicked. If the user credentials are valid, the administrator is taken to the main Web Interface page.

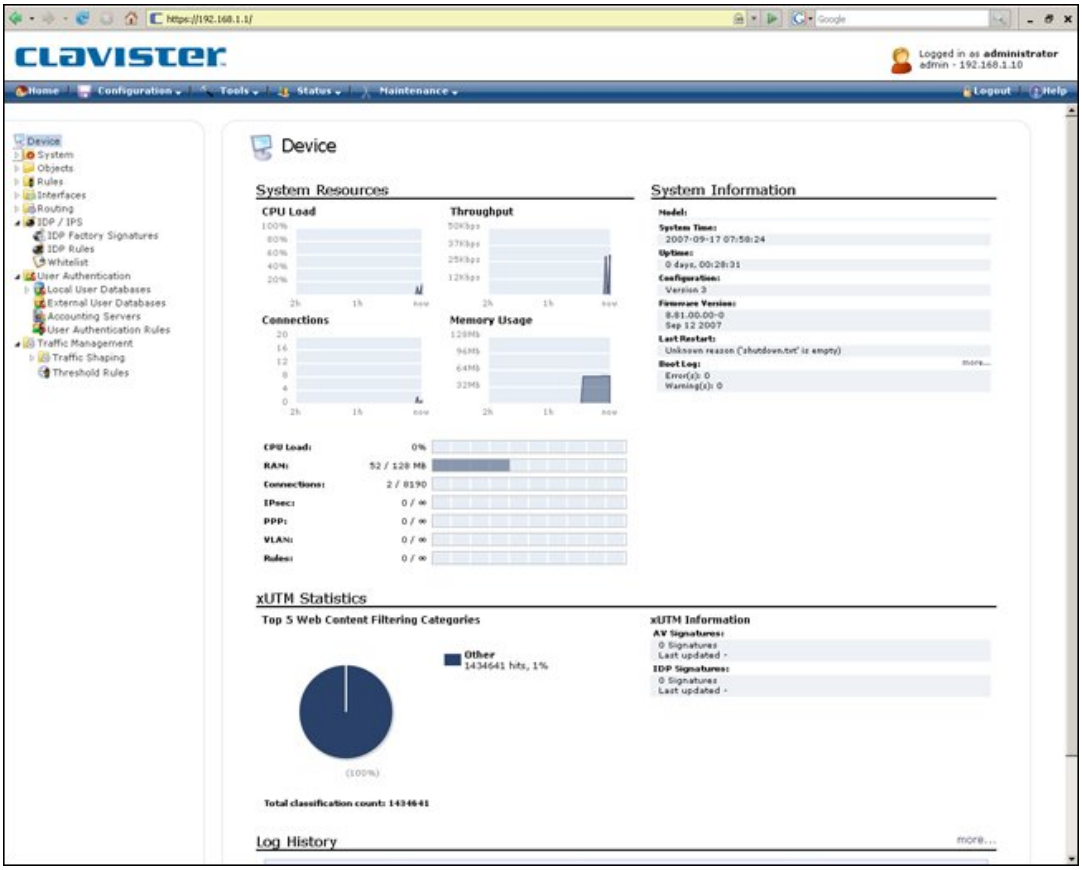
Multi-language Support

The WebUI login dialog offers the option to select a language other than English for the interface. Language support is provided by a separate resource files.

It may occasionally be the case that a CorePlus upgrade might contain features that temporarily lack a complete non-english translation because of time constraints. In this case the original english will be used as a temporary solution.

The Web Browser Interface

On the left hand side of the WebUI is a tree which allows navigation to the various CorePlus modules. The central area of the WebUI displays information about those modules. Current performance information is shown by default.



Note

Access to the Web Interface is regulated by the remote management policy. By default, the system will only allow web access from the internal network.

Interface Layout

The main Web Interface page is divided into three major sections:

Menu bar The menu bar located at the top of the Web Interface contains a number of buttons and drop-down menus that are used to perform configuration tasks as well as for navigation to various tools and status pages.

- **Home** - Navigates to the first page of the Web Interface.
- **Configuration**
  - **Save and Activate** - Saves and activates the configuration.
  - **Discard Changes** - Discards any changes made to the configuration during the current session.
  - **View Changes** - List the changes made to the configuration since it was last saved.
- **Tools** - Contains a number of tools that are useful for maintaining the system.
- **Status** - Provides various status pages that can be used for system diagnostics.
- **Maintenance**
  - **Update Center** - Manually update or schedule updates of the intrusion detection and antivirus signatures.
  - **License** - View license details or enter activation code.
  - **Backup** - Make a backup of the configuration to your local computer or restore a previously downloaded backup.
  - **Reset** - Restart the gateway or reset to factory default.
  - **Upgrade** - Upgrade the gateway's firmware.
  - **Technical support** - This option provides the option to download a file from the gateway which can be studied locally or sent to a technical support specialist to analyze a problem. This can be very useful since the information provided automatically includes many details that are required for troubleshooting.

Navigator The navigator located on the left-hand side of the Web Interface contains a tree representation of the system configuration. The tree is divided into a number of sections corresponding to the major building blocks of the configuration. The tree can be expanded to expose additional sections.

Main Window The main window contains configuration or status details corresponding to the section selected in the navigator or the menu bar.

Controlling Access to the Web Interface

By default, the Web Interface is accessible only from the internal network. If you need to enable access from other parts of the network, you can do so by modifying the remote management policy.

Example 3.1. Enabling remote management via HTTPS.

CLI

```
Device:/> add RemoteManagement RemoteMgmtHTTP https
     Network=all-nets Interface=any LocalUserDatabase=AdminUsers HTTPS=Yes
```

Web Interface

1. Go to **System > Remote Management > Add > HTTP/HTTPS Management**
2. Enter a **Name** for the HTTP/HTTPS remote management policy, for example *https*
3. Check the **HTTPS** checkbox
4. Select the following from the dropdown lists:
   - **User Database:** AdminUsers
   - **Interface:** any
   - **Network:** all-nets
5. Click **OK**

Caution

The above example is provided for informational purposes only. It is never recommended to expose any management interface to any user on the Internet.

Logging out from the Web Interface

When you have finished working in the Web Interface, you should always logout to prevent other users with access to your workstation to get unauthorized access to the system. Logout by clicking on the **Logout** button at the right of the menu bar.

Tip

*If there is a problem with the management interface when communicating alongside VPN tunnels, check the main routing table and look for an **all-nets** route to the VPN tunnel. If no specific route exists to the management interface then all management traffic coming from CorePlus will automatically be routed to the VPN tunnel. If this is the case then a route should be added by the administrator to route management traffic destined for the management network to the correct interface.*

3.1.4. The CLI

CorePlus provides a *Command Line Interface* (CLI) for administrators who prefer or require a command line approach to administration, or who need more granular control of system configuration. The CLI is available either locally through the serial console port (connection to this is described below), or remotely via an Ethernet interface using the *Secure Shell* (SSH) protocol from an SSH client.

The CLI provides a comprehensive set of commands that allow the display of configuration data as well as allowing runtime data to be displayed and allowing system maintenance tasks to be performed.

This section only provides a summary for using the CLI. For a complete reference for all CLI commands, see the separate Clavister *CLI Reference Guide*.

The most often used CLI commands are:

- *add* - Adds an object such as an IP address or a rule to a CorePlus configuration.
- *set* - Sets some property of an object to a value. For example, this might be used to set the source interface on an IP rule.
- *show* - Displays the current categories or display the values of a particular object.
- *delete* - Deletes a specific object.

CLI Command Structure

CLI commands usually begin with the structure: <*command*> <*object_type*> <*object_name*>. For example, to display an IP address object called *my_address*, the command would be:

```
Device:/> show Address IP4Address my_address
```

The second part of the command specifies the *object type* and is necessary to identify what category of object the object name refers to (consider that the same name might exist in two different categories).

Note
*The term **category** is sometimes referred to as the **context** of an object.*

A command like *add* can also include *object properties*. To add a new IP4Address object with an IP address of *10.49.02.01*, the command would be:

```
Device:/> add IP4Address my_address Address=10.49.02.01
```

The object *type* can be optionally preceded by the object *category*. A *category* groups together a set of *types* and mainly used with *tab completion* which is described below.

CLI Help

The CLI *help* command will show all available command options. A screen shot of the first part of the output from the *help* command is shown below:

```
CorePlus:/> help
Available commands (type "help help" for more help):

Set or show configuration data:
  activate  add  cc  delete  pskgen  reject  reset  set  show  undelete

Set or show runtime parameters:
  about        dhcp         igmp              natpool          shutdown
  alarm        dhcprelay    ikesnoop          netcon           sipalg
  arp          dhcpserver   ippool            ospf             sshserver
  arpsnoop     dns          ipsecglobalstats  pcapdump         stats
  ats          dnsbl        ipseckeepalive    pciscan          sysmsgs
  blacklist    dynroute     ipsecstats        pipes            techsupport
  buffers      frags        ipsectunnels      reconfigure      time
  cam          ha           killsa            routemon         uarules
  certcache    hostmon      languagefiles     routes           updatecenter
  cfglog       httpposter   license           rtmonitor        userauth
  connections  hwaccel      linkmon           rules            vlan
  cpuid        hwm          lockdown          services         vpnstats
  crashdump    idppipes     logout            sessionmanager
  dconsole     ifstat       memory            settings

Utilities:
  ping
```

Typing *help help* will give information about the *help* command itself.

The CLI Command History

Just like the console in many versions of Microsoft Windows, the up and down arrow keys allow the user to move through the list of commands in the *CLI command history*. For example, pressing the up arrow key once will make the last command executed appear at the current CLI prompt. After a command appears it can be re-executed in it's original form or changed first before execution.

Tab Completion

Remembering all the commands and their options can be difficult. CorePlus provides a feature called *tab completion* which means that pressing the tab key will cause automatically completion of the current part of the command. If completion is not possible then pressing the tab key will alternatively display the possible command options that are available.

Tab Completion of Data

A very useful feature with tab completion is the ability to automatically fill in the current values of data parameters in a command line. This is done by typing a period "." character followed by the tab key after the "=" character. For example, we may have typed the unfinished command:

```
set Address IP4Address lan_ip Address=
```

If we now type "." followed by a tab, CorePlus will display the current value for the *Address* parameter. If that value is, for example, *10.6.58.10* then the unfinished command line will automatically become:

```
set Address IP4Address lan_ip Address=10.6.58.10
```

CorePlus automatically inserts the current value of *10.6.58.10* and this can then be easily changed with the backspace or back arrow keys before completing the command.

In a similar way, the "<" character before a tab can be used to automatically fill in the default value for a parameter if no value has yet been set. For example:

```
add LogReceiverSyslog example Address=example_ip LogSeverity=< (tab)
```

Will fill in the default value for *LogSeverity*:

```
add LogReceiverSyslog example Address=example_ip
         LogSeverity=Emergency
```

However, if the "." character is used instead:

```
add LogReceiverSyslog example Address=example_ip LogSeverity=. (tab)
```

A list of all possible values is given:

```
add LogReceiverSyslog example Address=example_ip
     LogSeverity=Emergency,Alert,Critical,Error,Warning,Notice,Info
```

This list can then be edited with the back arrow and backspace keys.

Object Categories

It has been mentioned that objects are grouped by *type*, such as *IP4Address*. Types themselves are grouped by *category*. The type *IP4Address* belongs to the category *Address*. The main use of categories is in tab completion when searching for the right object type to use.

If a command such as *add* is entered and then the tab key is pressed, CorePlus displays all the available categories. By choosing a category and then pressing tab again all the object types for that category is displayed. Using categories means that the user has a simple way to specify what kind of object they are trying to specify and a manageable number of options are displayed after pressing tab.

Not all object types belong in a category. The object type *UserAuthRule* is a type without a category and will appear in the category list after pressing tab at the beginning of a command.

Selecting Object Categories

With some categories, it is necessary to first choose a member of that category with the *cc* (change category) command before individual objects can be manipulated. This is the case, for example, with routes. There can be more than one routing table, so when adding or manipulating a route we first have to use the *cc* command to identify which routing table we are interested in.

Suppose a route is to be added to the routing table *main*. The first command would be:

```
Device:/> cc RoutingTable main

Device:/main>
```

Notice that the command prompt changes to indicate the current category. We can now add the route:

```
Device:/main> add Route Name=new_route1 Interface=lan Network=lannet
```

To deselect the category, the command is *cc* on its own:

```
Device:/main> cc

Device:/>
```

The categories that require an initial *cc* command before object manipulation have a "/" character following their names when displayed by a *show* command. For example: *RoutingTable/*.

Specifying Multiple Property Values

Sometimes a command property may need multiple values. For example, some commands use the property *AccountingServers* and more than one value can be specified for this property. When specifying multiple values, they should be separated by a comma "," character. For example, if three servers *server1*, *server2*, *server3* need to be specified then the property assignment in the command would be:

```
AccountingServers=server1,server2,server3
```

Inserting into Rule Lists

Rule lists such as the IP rule set have an ordering which is important. When adding using the CLI *add* command, the default is to add a new rule to the end of a list. When placement at a particular position is crucial, the *add* command can include the *Index=* parameter as an option. Inserting at the first position in a list is specified with the parameter *Index=1* in an *add* command, the second position with the parameter *Index=2* and so on.

Referencing by Name

The naming of some objects is optional and is done with the *Name=* parameter in an *add* command. An object, such as a threshold rule, will always have an *Index* value which indicates its position in the rule list but can optionally be allocated a name as well. Subsequent manipulation of such a rule can be done either by referring to it by its index, that is to say its list position, or by alternatively using the name assigned to it.

The *CLI Reference Guide* lists the parameter options available for each CorePlus object, including the *Name=* and *Index=* options.

Using Unique Names

For convenience and clarity, it is recommended that a name is assigned to all objects so that it can be used for reference if required. Reference by name is particularly useful when writing CLI scripts.

The CLI will enforce unique naming within an object type. For reasons of backward compatibility to earlier CorePlus releases, an exception exists with IP rules which can have duplicate names, however it is strongly recommended to avoid this. If a duplicate IP rule name is used in two IP rules then only the *Index* value can uniquely identify each IP rule in subsequent CLI commands. Referencing an IP rule with a duplicated name will fail and result in an error message.

Using Hostnames in the CLI

For certain CLI commands, IP addresses can optionally be specified as a textual hostname instead an IP4Address object or raw IP address such as *192.168.1.10*. When this is done, the hostname must be prefixed with the letters *dns:* to indicate that a DNS lookup must be done to resolve the hostname to an IP address. For example, the hostname *host.company.com* would be specified as *dns:host.company.com* in the CLI.

The parameters where URNs might be used with the CLI are:

- The *Remote Endpoint* for IPsec, L2TP and PPTP tunnels.
- The *Host* for LDAP servers.

When DNS lookup needs to be done, at least one public DNS server must be configured in CorePlus for hostnames to be translated to IP addresses.

Serial Console CLI Access

The serial console port is a local RS-232 port on the Clavister Security Gateway that allows direct access to the CorePlus CLI through a serial connection to a PC or dumb terminal. To locate the serial console port on your Clavister hardware, see the corresponding hardware installation guide.

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

- A terminal or a computer with a serial port and the ability to emulate a terminal (such as using the *Hyper Terminal* software included in some Microsoft Windows editions). The serial console port uses the following default settings: *9600 bps, No parity, 8 stop bits and 1 stop bit.*
- A RS-232 cable with appropriate connectors. An appliance package includes a 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.

Changing the Serial Console Line Speed

The console line speed can be changed either through the WebUI or through the CLI command. The new speed will only come into effect after restarting CorePlus.

Example 3.2. Setting the Console Line Speed

In this example the console line speed is set to 19200 bps.

CLI

```
Device:/> set COMPortDevice COM1 BitsPerSecond=19200
```

Web Interface

Go to **System > Com Port Devices**

Click the console port line, configure the speed and then click **OK**

SSH (Secure Shell) CLI Access

The SSH (Secure Shell) protocol can be used to access the CLI over the network from a remote host. SSH is a protocol primarily used for secure communication over insecure networks, providing strong authentication and data integrity. SSH clients are freely available for almost all hardware platforms.

CorePlus supports version 1, 1.5 and 2 of the SSH protocol. SSH access is regulated by the remote management policy in CorePlus, and is disabled by default.

Example 3.3. Enabling SSH Remote Access

This example shows how to enable remote SSH access from the *lannet* network through the *lan* interface by adding a rule to the remote management policy.

CLI

```
Device:/> add RemoteManagement RemoteMgmtSSH ssh Network=lannet Interface=lan
     LocalUserDatabase=AdminUsers
```

Web Interface

1. Go to **System > Remote Management > Add > Secure Shell Management**
2. Enter a **Name** for the SSH remote management policy, for example *ssh_policy*
3. Select the following from the dropdown lists:
   - **User Database:** AdminUsers
   - **Interface:** lan
   - **Network:** lannet
4. Click **OK**

Logging on to the CLI

When access to the CLI has been established to CorePlus through the serial console or an SSH client, the administrator will 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 auditing.

When accessing the CLI, the system will respond with a login prompt. Enter your username and press *Enter*, followed by your password and then *Enter* again.

After a successful logon the command prompt will appear:

```
Device:/>
```

If a welcome message has been set then it will be displayed directly after the logon. For security reasons, it is advisable to disable or anonymize the CLI welcome message.

Changing the CLI Prompt

The default CLI prompt is:

```
Device:/>
```

This can be customized, for example, to *gw-world:/>*, by using the CLI command:

```
Device:/> set device name="gw-world"
```

The *CLI Reference Guide* uses the command prompt *gw-world:/>* throughout.

Note
When the command line prompt is changed to a new string value, this string also appears as the new device name in the top level node of the WebUI tree-view.

Activating and Committing Changes

If any changes are made to the current configuration through the CLI, those changes will not be uploaded to CorePlus until the command:

```
Device:/> activate
```

is issued. Immediately following the *activate* command, the command:

```
Device:/> commit
```

should be issued to make those changes permanent. If a *commit* command is not issued within a default time period of 30 seconds then the changes are automatically undone and the old configuration restored.

Explicity Checking Configuration Integrity

After changing a configuration on the Clavister Security Gateway, and before the *activate*/*commit* commands, it is possible to explicitly check for any problems in a configuration using the command:

```
Device:/> show -errors
```

This will cause CorePlus to scan the configuration about to be activated and list any problems. A possible problem that might be found in this way is a reference to an IP object in the Address Book that does not exist in a restored configuration backup.

Logging off from the CLI

After finishing working with the CLI, it is recommended to logout in order to avoid letting anyone getting unauthorized access to the system. Log off by using the *exit* or the *logout* command.

3.1.5. CLI Scripts

To allow the administrator to easily store and execute sets of CLI commands, CorePlus provides a feature called *CLI scripting*. A *CLI script* is a predefined sequence of CLI commands which can be executed after they are saved to a file and the file is then uploaded to the Clavister Security Gateway.

The steps for creating a CLI script are as follows:

1. Create a text file with a text editor containing a sequential list of CLI commands, one per line. The Clavister recommended convention is for these files to use the file extension *.sgs* (S*ecurity* G*ateway* S*cript*). The filename (including the extension) should not be more than 16 characters.

2. Upload the file to the Clavister Security Gateway using Secure Copy (SCP). Script files must be stored in a directory under the root called */scripts*. SCP uploading is discussed in detail in Section 3.1.6, “Secure Copy”.

3. Use the CLI command *script -execute* to run the script file.

The CLI *script* command is the tool used for script management and execution. The complete syntax of the command is described in the *CLI Reference Guide* and specific examples of usage are detailed in the following sections. See also Section 3.1.4, “The CLI” in this manual.

Executing Scripts

As mentioned above, the *script -execute* command launches a named script file that has been previously uploaded to the Clavister Security Gateway. For example, to execute the script file *my_script.sgs* which has already been uploaded, the CLI command would be:

```
Device:/> script -execute -name=my_script.sgs
```

Script Variables

A script file can contain any number of *script variables* which are called:

$1, $2, $3, $4.....$n

The values substituted for these variable names are specified as a list at the end of the *script -execute* command line. The number *n* in the variable name indicates the variable value's position in this list. *$1* comes first, *$2* comes second and so on.

Note: $0 is reserved

Notice that the name of the first variable is $1. The variable $0 is reserved and is always replaced before execution by the name of the script file itself.

For example, a script called *my_script.sgs* is to be executed with IP address *126.12.11.01* replacing all occurrences of *$1* in the script file and the string *If1 address* replacing all occurrences of *$2*.

The file *my_script.sgs* contains the single CLI command line:

```
add IP4Address If1_ip Address=$1 Comments=$2
```

To run this script file after uploading, the CLI command would be:

```
:/> script -execute -name=my_script.sgs 126.12.11.01 "If1 address"
```

When the script file runs, the variable replacement would mean that the file becomes:

```
add IP4Address If1_ip Address=126.12.11.01 Comments="If1 address"
```

Error Handling

If an executing CLI script file encounters an error condition, the default behavior is for the script to terminate. This behavior can be overridden by using the *-force* option. To run a script file called *my_script2.sgs* in this way, the CLI command is:

```
Device:/> script -execute -name=my_script2.sgs -force
```

If *-force* is used, the script will continue to execute even if errors are returned by a command in the script file.

Script Output

Any output from script execution will appear at the CLI console. Normally this output only consists of any error messages that occur during execution. To see the confirmation of each command completing, the *-verbose* option should be used:

```
Device:/> script -execute -name=my_script2.sgs -verbose
```

Saving Scripts

When a script file is uploaded to the Clavister Security Gateway, it is initially kept only in temporary RAM memory. If CorePlus restarts then any uploaded scripts will be lost from this volatile memory and must be uploaded again to run. To store a script between restarts, it must explicitly be moved to non-volatile CorePlus *disk* memory by using the *script -store* command.

To move the example *my_script.sgs* to non-volatile memory the command would be:

```
Device:/> script -store -name=my_script.sgs
```

Alternatively, all scripts can be moved to non-volatile memory with the command:

```
Device:/> script -store -all
```

Listing Scripts

The *script* on its own, command without any parameters, lists all the scripts currently available and indicates the size of each script as well as the type of memory where it resides (residence in non-volatile memory is indicated by the word *Disk* in the *Memory* column).

```
Device:/> script
```

```
Name            Storage       Size (bytes)
--------------  ------------  --------------
my_script.sgs   RAM           8
my_script2.sgs  Disk          10
```

To list the content of a specific uploaded script file, for example *my_script.sgs* the command would be:

```
Device:/> script -show -name=my_script.sgs
```

Creating Scripts Automatically

When the same configuration objects needs to be copied between multiple Clavister Security Gateways, then one way to do this with the CLI is to create a script file that creates the required objects and then upload to and run the same script on each device.

If we already have a CorePlus installation that already has the objects configured that need to be copied, then running the *script -create* command on that installation provides a way to automatically create the required script file. This script file can then be downloaded to the local management workstation and then uploaded to and executed on other Clavister Security Gateways to duplicate the objects.

For example, suppose the requirement is to create the same set of **IP4Address** objects on several Clavister Security Gateways that already exist on a single unit. The administrator would connect to the single unit with the CLI and issue the command:

```
Device:/> script -create Address IP4Address -name new_script.sgs
```

This creates a script file called *new_script_sgs* which contains all the CLI commands necessary to create all **IP4Address** address objects in that unit's configuration. The created file's contents might, for example, be:

```
add IP4Address If1_ip Address=10.6.60.10
add IP4Address If1_net Address=10.6.60.0/24
add IP4Address If1_br Address=10.6.60.255
add IP4Address If1_dns1 Address=141.1.1.1
                "
                "
                "
```

The file *new_script_sgs* can then be downloaded with SCP to the local management workstation and then uploaded and executed on the other Clavister Security Gateways. The end result is that all units will have the same **IP4Address** objects in their address book.

The name of the file created using the *-create* option cannot be greater than 16 characters in length (including the extension) and the filetype should be *.sgs*.

Tip

*To list the created CLI commands on the console instead of saving them to a file, leave out the option **-name=** in the **script -create** command.*

Certain aspects of a configuration which are hardware dependent cannot have a script created using the *-create* option. This is true when the CLI node type in the *script -create* command is one of:

COMPortDevice
Ethernet
EthernetDevice
Device

If one of these node types is used then the error message *script file empty* is returned by CorePlus.

Commenting Script Files

Any line in a script file that begins with the # character is treated as a comment. For example:

```
# The following line defines the If1 IP address
add IP4Address If1_ip Address=10.6.60.10
```

Scripts Running Other Scripts

It is possible for one script to run another script. For example, the script *my_script.sgs* could contain the line:

```
"
"
script -execute -name my_script2.sgs
"
"
```

CorePlus allows the script file *my_script2.sgs* to execute another script file and so on. The maximum depth of this script nesting is 5.

3.1.6. Secure Copy

To upload and download files to or from the Clavister Security Gateway, the *secure copy* (SCP) protocol can be used. SCP is based on the SSH protocol and many freely available SCP clients exist for almost all platforms. The command line examples below are based on the most common command format for SCP client software.

SCP Command Format

SCP command syntax is straightforward for most console based clients. The basic command used here is *scp* followed by the source and destination for the file transfer.

Upload is done with the command:

```
> scp <local_filename> <destination_gateway>
```

Download is done with the command:

```
> scp <source_gateway> <local_filename>
```

The source or destination Clavister Security Gateway is of the form: *<user_name>@<gateway_ip_address>:<filepath>*. For example: *admin@10.62.11.10:config.bak*. The *<user_name>* must be a defined CorePlus user in the administrator user group.

Note on the password prompt

SCP will normally prompt for the user password after the command line but that prompt is not shown in the examples that follow.

The following table summarizes the operations that can be performed between an SCP client and CorePlus.

File type	Upload possible	Download possible
Configuration Backup (config.bak)	Yes (also with WebUI)	Yes (also with WebUI)
System Backup (full.bak)	Yes (also with WebUI)	Yes (also with WebUI)
Firmware upgrades	Yes	No
Licenses (license.lic)	Yes (also with WebUI)	No
Certificates	Yes	No
SSH public keys	Yes	No
Web auth banner files	Yes	Yes
Web content filter banner files	Yes	Yes

CorePlus File organization

CorePlus maintains a simple 2 level directory structure which consists of the top level *root* and a number of sub-directories. However, these "directories" such as *sshlclientkey* should be more correctly thought of as *object types*. All the files stored in the CorePlus root as well as all the object types can be displayed using the CLI command *ls*.

The resulting output is shown below:

```
Device:/> ls

HTTPALGBanners/
HTTPAuthBanners/
certificate/
config.bak
full.bak
license.lic
script/
sshclientkey/
```

Aprt from the individual files, the objects types listed are:

- *HTTPALGBanners/* - The banner files for user authentication HTML. Uploading these is described further in Section 7.3.4.4, “Customizing HTML Pages”.
- *HTTPAuthBanner/* - The banner files for HTML ALG dynamic content filtering. Uploading these is described further in Section 7.3.4.4, “Customizing HTML Pages”.
- *certificate/* - The object type for all digital certificates.
- *script/* - The object type for all CLI scripts. Scripts are described further in Section 3.1.5, “CLI Scripts”.
- *sshclientkey/* - The SSH client key object type.

Examples of Uploading and Downloading

In some cases, a file is located in the CorePlus root. The license file (*license.lic*) falls into this category, as well as backup files for configurations (*config.bak*) and the complete system (*full.bak*). When uploading, these files contain a unique header which identifies what they are. CorePlus checks this header and ensures the file is stored only in the root (all files do not have a header).

If an administrator username is *admin1* and the IP address of the Clavister Security Gateway is *10.5.62.11* then to upload a configuration backup, the SCP command would be:

```
> scp config.bak admin1@10.5.62.11:
```

To download a configuration backup to the current local directory, the command would be:

```
> scp admin1@10.5.62.11:config.bak ./
```

To upload a file to an object type under the root, the command is slightly different. If we have a local CLI script file called *my_script.sgs* then the upload command would be:

```
> scp my_script.sgs admin1@10.5.62.11:script/
```

If we have the same CLI script file called *my_scripts.sgs* stored on the Clavister Security Gateway then the download command would be:

```
> scp admin1@10.5.62.11:script/my_script.sgs ./
```

Activating Uploads

Like all configuration changes, SCP uploads only become active after the CLI commands *activate* have been issued and this must be followed by *commit* to make the change permanent.

Uploads of firmware upgrades (packaged in *.upg* files) or a full system backup (*full.bak*) are the exception. Both of these file types will result in an automatic system reboot. The other exception is for script uploads which do not affect the configuration.

3.1.7. The Console Boot Menu

The Clavister *firmware loader* is the base software on top of which CorePlus runs and the administrator's direct interface to this loader is called the *console boot menu*. This section discusses the boot menu and also how the serial console can be used to issue CorePlus CLI commands.

The boot menu is only accessible from the serial console located on the hardware chassis, and is only available if CorePlus has been shutdown or not yet started and power is on to the hardware.

Initial Menu without a Password Set

After powering up the Clavister Security Gateway, there is a five second interval before CorePlus starts up, and in that time the message *Press any key to abort and load boot menu* is displayed on the console. If any console key is pressed during these 5 seconds then CorePlus startup pauses and the following menu is displayed:

- **Start System**
  This initiates the startup of CorePlus.
- **Display Latest Shutdown Message**
  This shows the last console message shown before system shutdown.
- **Display Latest Crash Dump Message**
  This shows the latest crash dump message caused by a CorePlus problem.
- **Reset Options**
  This displays a sub-menu of reset options which are described below.
- **Enable Console Password**
  Set a password for console access. Until a password is set, anyone can utilize the console so selecting this option is recommended. The console will prompt for a password and ask for confirmation.

Options after Setting a Password

After a password is set with the **Enable Console Password** step above, the following menu options are displayed (and this menu will be displayed after a successful logon):

- **Start System**
  This initiates the startup of CorePlus.
- **Display Latest Shutdown Message**
  This shows the last console message shown before system shutdown.
- **Display Latest Crash Dump Message**
  This shows the latest crash dump message caused by a CorePlus problem.
- **Reset Options**
  This displays a sub-menu of reset options which are described below.
- **Change Console Password**
  This option allows the administrator to change the current password for console access and it is recommended this is set as soon as possible. If it is not set, anyone can gain access to the local console port to reconfigure CorePlus.

  It is important to understand that this password is not related to the username and password used for login through user interfaces available remotely such as the WebUI. The console password is used only with a console connected directly through the local RS232 port and there is no username associated with it.

Initial Menu with a Password Set

Once a password is set by the administrator, the initial menu that is displayed on interrupting startup will become altered so that it contains the following options:

- **Login**
- **Start System**
  This initiates the startup of CorePlus.
- **Display Latest Shutdown Message**
  This shows the last console message shown before system shutdown.
- **Display Latest Crash Dump Message**
  This shows the latest crash dump message caused by a CorePlus problem.

The Reset Menu

The *Reset Options* menu offers the following choices:

- **Reset to Factory Defaults**
  This option will restore the hardware to its initial factory state. The operations performed if this option is selected are the following:
  - Remove the CorePlus configuration files. This includes all certificates as well as HA and DHCP lease settings.
  - Reset the loader settings.
  - Remove console security so there is no console password.

  - Restore default CorePlus and loader executables. If the hardware was delivered as a CorePlus 8.nn system then the WebUI will no longer function and the upgrade wizard will have to be re-run.

  - The license will NOT be removed.

- **Reset to Base Configuration**
  This will only reset the configuration to be the original, default CorePlus configuration file. Other options, such as console security, will not be affected.

- **Transfer System**
  This option is for software only installations running on non-Clavister hardware. It makes it possible to transfer (or copy) an image of the complete CorePlus system to another portable media, such as a USB stick. Booting can then be done using this media on another computer so that the CorePlus system can be installed there on local disk with another "Transfer System" operation.

- **Return to main menu** - Go back to the previous, main menu.

Note: A reset does not delete all files
Resetting to factory defaults will not delete any auxiliary files that have been created in the Clavister Security Gateway memory. For example, any files created by the ***pcapdump*** *command will not be deleted. Files related to Anti-Virus or IDP also will not be deleted. If these auxiliary files need to be deleted then this must be done explicitly through the appropriate CLI command.*

Using the Console for CLI Commands

CorePlus will start up either because the initial start up sequence wasn't interrupted to enter the boot menu or because start up was commenced from the boot menu. Once CorePlus is up and running, the serial console can also be used to enter any CLI command in the same way that a SSH client can.

If a console password has been set and a login has not yet been performed then CorePlus will prompt for the console password before CLI commands can be entered and it is therefore recommended the console password is set as soon as possible.

Note: Output buffer limitations
The only limitation with issuing CLI commands through the serial console is that there is a finite buffer allocated for output. This buffer limit means that a large volume of console output may be truncated. This happens rarely and usually only with data dumps from certain diagnostic commands. In such cases it is better to issue the commands using an SSH client instead.

3.1.8. Management Advanced Settings

Under the **Remote Management** section of the WebUI a number of advanced settings can be found. These are:

SSH Before Rules

Enable SSH traffic to the security gateway regardless of configured IP Rules.

Default: *Enabled*

WebUI Before Rules

Enable HTTP(S) traffic to the security gateway regardless of configured IP Rules.

Default: *Enabled*

Local Console Timeout

Number of seconds of inactivity until the local console user is automatically logged out.

Default: *900*

NetCon Idle Timeout

Number of seconds of inactivity until the Netcon session is close.

Default: *600*

NetCon Before Rules

For NetCon traffic to CorePlus: add an invisible rule to the IP rule set to allow Netcon traffic. If this setting is disabled then NetCon traffic will be subject to the IP rule set like any other traffic.

Default: *Enabled*

NetConMaxChannels

For NetCon traffic to CorePlus the following number of connections are guaranteed for each connection type:

- Consoles: 4
- Realtime loggers: 4
- Stat pollers: 4
- Receive contexts: 2
- Send contexts: 4

NetConMaxChannels is the maximum total allowed for all these connection types. This means that with the default value of 18, 2 extra connections are allowed and they can be of any of the above types.

Default: *18*

Validation Timeout

Specifies the amount of seconds to wait for the administrator to log in before reverting to the previous configuration.

Default: *30*

WebUI HTTP port

Specifies the HTTP port for the Web Interface.

Default: *80*

WebUI HTTPS port

Specifies the HTTP(S) port for the Web Interface.

Default: *443*

HTTPS Certificate

Specifies which certificate to use for HTTPS traffic. Only RSA certificates are supported.

Default: *HTTPS*

3.1.9. Working with Configurations

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 on. 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 Web Interface the configuration objects are organized into a tree-like structure based on the type of the object.

In the CLI similar configuration object types are grouped together in a *category*. These categories are different from the structure used in the Web Interface to allow quick access to the configuration objects in the CLI. The IP4Address, IP4Group and EthernetAddress types are, for instance, grouped in a category named *Address*, as they all represent different addresses. Consequently, Ethernet and VLAN objects are all grouped in a category named *Interface*, as they are all interface objects. The categories have actually no impact on the system configuration; they are merely provided as means to simplify administration.

The following examples show how to manipulate objects.

Example 3.4. Listing Configuration Objects

To find out what configuration objects exist, you can retrieve a listing of the objects. This example shows how to list all service objects.

CLI

```
Device:/> show Service
```

A list of all services will be displayed, grouped by their respective type.

Web Interface

1. Go to **Objects > Services**
2. A web page listing all services will be presented.

A list contains the following basic elements:

- **Add Button** - Displays a dropdown menu when clicked. The menu will list all types of configuration items that can be added to the list.
- **Header** - The header row displays the titles of the columns in the list. The tiny arrow images next to each title can be used for sorting the list according to that column.

- **Rows** - Each row in the list corresponds to one configuration item. Most commonly, each row starts with the name of the object (if the item has a name), followed by values for the columns in the list.

A single row in the list can be selected by clicking on the row on a spot where there is no hyperlink. The background color of the row will turn dark blue. Right-clicking the row will display a menu where you can choose to edit or delete the object as well as modify the order of the objects.

Example 3.5. Displaying a Configuration Object

The simplest operation on a configuration object is to show its contents, in other words the values of the object properties. This example shows how to display the contents of a configuration object representing the *telnet* service.

CLI

```
Device:/> show Service ServiceTCPUDP telnet

          Property  Value
 ----------------  -------
             Name:  telnet
 DestinationPorts:  23
             Type:  TCP
      SourcePorts:  0-65535
         SYNRelay:  No
   PassICMPReturn:  No
              ALG:  (none)
      MaxSessions:  1000
         Comments:  Telnet
```

The Property column lists the names of all properties in the ServiceTCPUDP class and the Value column lists the corresponding property values.

Web Interface

1. Go to **Objects > Services**
2. Click on the **telnet** hyperlink in the list
3. A web page displaying the telnet service will be presented

Example 3.6. 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. This example shows how to edit the *Comments* property of the *telnet* service.

CLI

```
Device:/> set Service ServiceTCPUDP telnet Comments="Modified Comment"
```

Show the object again to verify the new property value:

```
Device:/> show Service ServiceTCPUDP telnet

          Property  Value
 ----------------  -------
             Name:  telnet
 DestinationPorts:  23
             Type:  TCP
      SourcePorts:  0-65535
         SYNRelay:  No
   PassICMPReturn:  No
              ALG:  (none)
      MaxSessions:  1000
         Comments:  Modified Comment
```

Web Interface

1. Go to **Objects > Services**
2. Click on the **telnet** hyperlink in the list
3. In the **Comments** textbox, enter your new comment
4. Click **OK**

Verify that the new comment has been updated in the list.

Important

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

Example 3.7. Adding a Configuration Object

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

CLI

```
Device:/> add Address IP4Address myhost Address=192.168.10.10
```

Show the new object:

```
Device:/> show Address IP4Address myhost

              Property  Value
 --------------------  -------------
                  Name:  myhost
               Address:  192.168.10.10
       UserAuthGroups:  (none)
 NoDefinedCredentials:  No
              Comments:  (none)
```

Web Interface

1. Go to **Objects > Address Book**
2. Click on the **Add** button
3. In the dropdown menu displayed, select **IP Address**
4. In the **Name** text box, enter *myhost*
5. Enter 192.168.10.10 in the **IP Address** textbox
6. Click **OK**
7. Verify that the new IP4 address object has been added to the list

Example 3.8. Deleting a Configuration Object

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

CLI

```
Device:/> delete Address IP4Address myhost
```

Web Interface

1. Go to **Objects > Address Book**
2. Right-click on the row containing the **myhost** object
3. In the dropdown menu displayed, select **Delete**

The row will be rendered with a strike-through line indicating that the object is marked for deletion.

Example 3.9. Undeleting a Configuration Object

A deleted object can always be restored until the configuration has been activated and committed. This example shows how to restore the deleted IP4Address object shown in the previous example.

CLI

```
Device:/> undelete Address IP4Address myhost
```

Web Interface

1. Go to **Objects > Address Book**
2. Right-click on the row containing the **myhost** object
3. In the dropdown menu displayed, select **Undo Delete**

Listing Modified Objects

After modifying several configuration objects, you might want to see a list of the objects that were changed, added and removed since the last commit.

Example 3.10. Listing Modified Configuration Objects

This example shows how to list configuration objects that have been modified.

CLI

```
Device:/> show -changes

    Type           Object
    -------------  ------
 -  IP4Address     myhost
 *  ServiceTCPUDP  telnet
```

A "+" character in front of the row indicates that the object has been added. A "*" character indicates that the object has been modified. A "-" character indicates that the object has been marked for deletion.

Web Interface

1. Go to **Configuration > View Changes** in the menu bar

A list of changes is displayed

Activating and Committing a Configuration

After 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 affect 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 a lost connection could not be re-established then CorePlus will revert to using the previous configuration. This is a fail-safe mechanism and, amongst others things, can help prevent a remote administrator from locking themselves out.

Example 3.11. Activating and Committing a Configuration

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

CLI

```
Device:/> activate
```

The system will validate and start using the new configuration. When the command prompt is shown again:

```
Device:/> commit
```

The new configuration is now committed.

Web Interface

1. Go to **Configuration > Save and Activate** in the menu bar
2. Click **OK** to confirm

The web browser will automatically try to connect back to the Web Interface after 10 seconds. If the connection succeeds, this is interpreted by CorePlus that remote management is still working. The new configuration is then automatically committed.

Note

The configuration must be committed before changes are saved. All changes to a configuration can be ignored simply by not committing a changed configuration.

3.2. Events and Logging

3.2.1. Overview

The ability to log and analyze system activities is an essential feature of CorePlus. Logging enables not only monitoring of system status and health, but also allows auditing of network usage and assists in trouble-shooting.

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

Log events are always generated for certain aspects of CorePlus such as buffer usage, DHCP clients, High Availability and IPsec. The generation of events for other CorePlus subsystems such as DHCP Relay, DHCP Servers and IP Rules can be enabled as needed.

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 per Clavister Security Gateway, with each receiver having its own customizable event filter.

3.2.2. 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 example, 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 format, with attributes that include category, severity and recommended actions. These attributes enable easy filtering of messages, either within CorePlus prior to sending to an event receiver, or as part of the analysis after logging and storing messages on an external log server.

A list of all event messages can be found in the CorePlus *Log Reference Guide*. That guide also describes the design of event messages, the meaning of severity levels and the various attributes available. The severity of each event is predefined and it can be, in order of severity, one of:

Emergency
Alert
Critical
Error
Warning
Notice
Info
Debug

By default all messages of level **Info** and above are sent. The **Debug** category of designed for troubleshooting only and should only be turned on if required to try and solve a problem. Messages of all severity levels are found listed in the CorePlus *Log Reference Guide*.

3.2.3. Event Message Distribution

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

CorePlus can distribute event messages in the following ways:

Memlog	CorePlus has a built in logging mechanism known as the *Memory Log*. This retains all event log messages in memory and allows direct viewing of log messages through the Web Interface.
Syslog	The de-facto standard for logging events from network devices. If other network devices are already logging to Syslog servers, using syslog with CorePlus messages can simplify overall administration.
FWLog	The Clavister proprietary format for logging event messages, the FWLog format has a high level of detail and is suitable for analyzing large amounts of log data.
SNMP Traps	Commonly used to collect and respond to critical alerts from network devices.
NetCon Real-time Log	Event messages will always be distributed to any Real-time Log listeners that are connecting using the NetCon protocol.

3.2.3.1. Logging to Memlog

Memlog is an optional CorePlus feature that allows logging direct to memory in the Clavister Security Gateway instead of sending messages to an external server. Memlog messages can be examined through the standard user interfaces.

The Memlog memory is limited to a fixed predetermined size since hardware resources are limited. When the allocated memory is filled up with log messages, the oldest messages are discarded to make room for newer incoming messages. This means that MemLog holds a limited number of messages since the last system initialization and once the buffer fills they will only be the most recent. This means that when CorePlus is creating large numbers of messages in systems with, for example, large numbers of VPN tunnels, the Memlog information becomes less meaningful since it reflects a limited recent time period.

3.2.3.2. Logging to Syslog Hosts

Overview

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

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

Message Format

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.

Note

*The **Prio=** field in SysLog messages contains the same information as the **Severity** field for Clavister Logger messages, however the ordering of the numbering is reversed.*

Example 3.12. 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:

CLI

```
Device:/> add LogReceiverSyslog my_syslog IPAddress=195.11.22.55
```

Web Interface

1. Go to **System > Log and Event Receivers > Add > Syslog Receiver**
2. Specify a suitable name for the event receiver, for example *my_syslog*
3. Enter *195.11.22.55* as the **IP Address**
4. Select an appropriate facility from the **Facility** list - the facility name is commonly used as a filter parameter in most syslog daemons.
5. 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: Syslog server configuration

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.

Logging and Analysis with InSight™

As a complement to the standard monitoring and analysis tools such as MemLog, Clavister offers a separately licensed product called *InSight*. This product consists of a combination of a SysLog server plus a client for detailed analysis of events messages sent to the server.

InSight functionality goes far beyond the analysis capabilities normally available with CorePlus and is aimed at administrators who require a more advanced tool for examining log messages. Clavister InSight can be downloaded for trial from the Clavister Customer Web. A separate dedicated manual describes the product in detail and this is included with the software as well as being available on the Clavister website.

3.2.3.3. Logging to the Clavister Logger

The *Clavister Logger* is a proprietary Clavister logging product that uses the proprietary Clavister *FWLog* message format for sending and storing log data. This logger is also referred to as the *FWLog Receiver*.

For backwards compatibility, CorePlus versions older than 8.90 support output to this logger but the software itself is not included with the distribution packet since analysis of logger output is only possible through older versions of CorePlus.

Example 3.13. Enabling Logging to Clavister Loggers

To enable logging of all events with a severity greater than or equal to *Notice* to the Clavister Logger (*FWLog Receiver*) with IP address *195.11.22.55* listening on port *999* (the default), follow the steps below:

CLI

```
Device:/> add LogReceiver LogReceiverFWLog my_fwlog IPAddress=195.11.22.55 Port=999
```

Web Interface

1. Go to **System > Log and Event Receivers > Add > FWLog Receiver**
2. Specify a suitable name for the event receiver, for example *my_fwlog*
3. Enter *195.11.22.55* as the **IP Address**
4. 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*.

3.2.3.4. 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 file *Clavister-TRAP.MIB* which is included under the *SNMP* directory in the CorePlus distribution, defines the SNMP objects and data types that are used to describe an SNMP Trap received from CorePlus.

There is one generic trap object called *OSGenericTrap*, 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 3.14. 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:

CLI

```
Device:/> add LogReceiver EventReceiverSNMP2c my_snmp IPAddress=195.11.22.55
```

Web Interface

1. Go to **Log & Event Receivers > Add > SNMP2cEventReceiver**
2. Specify a name for the event receiver, for example *my_snmp*
3. Enter 195.11.22.55 as the **IP Address**
4. Enter an SNMP **Community String** if needed by the trap receiver
5. 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.

3.2.4. Customizing Log Messages

The sending of log messages can be customized to meet specific needs. This is done on a per log receiver basis. There are two types of customization that can be performed for a log receiver:

- Specific log messages or categories of log messages may be excluded from sending.
- The default severity of specific log messages or categories of log messages can be changed.

Log Message Categories

A log message is identified by a unique number. The first 3 digits of that number identify the *Category* to which the log message belongs. Categories relate to specific subsystems in CorePlus. For example, IPsec is such a subsystem and is identified by the category ID *018*. The CorePlus *Log Reference Guide* details all such log messages along with their unique ID number identifiers.

Changing the Default Severity Level

A log message has a default severity level, based on the importance of the event that caused the log message to be sent. However, this might differ between different installations. An event that is crucial in one scenario might not be of interest in another. For this reason it is possible to change the default severity level for a particular log message or for an entire category of messages.

Excluding a Log Message

A particular log message or category of messages can be excluded so that they will never be sent. This is useful if an event that is frequent causes a log message to be sent too often when it is of no interest to the administrator. An entire event category may similarly be of no interest to the administrator. The event ID or category can be identified from the CorePlus *Log Reference Guide*.

The Include/Exclude Mechanism

For each log receiver definition, a list of event and/or category *Includes* as well as *Excludes* can be created to define the messages that are to be sent or not sent to the receiver.

A specific message or messages may be included whereas the group is excluded. This means that only the included messages will be sent from the category. Likewise, only specific messages might be excluded from a category.

3.2.5. Advanced Log Settings

The following advanced settings for logging are available to the administrator:

Send Limit

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.

A 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* (once per hour)

Alarm Repetition Interval

The delay in seconds between alarms when a continuous alarm is used. Minimum 0, Maximum 10,000.

Default: *60* (one minute)

3.3. RADIUS Accounting

3.3.1. 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.

RADIUS Architecture

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 security 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 9.2, “Authentication Setup”).

3.3.2. 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.

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 signalling 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 1st January, 1970. 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, two more attributes may be 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.

3.3.3. 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.

3.3.4. 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.

3.3.5. 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.

3.3.6. RADIUS Accounting and High Availability

In an HA cluster, accounting information is synchronized 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 problem with accounting information synchronization could occur if an active unit has an authenticated user for whom the associated connection times out before it is synchronized on 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.

3.3.7. 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 *Allow on error* 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 affected user will automatically be logged out even if they have already been authenticated.

3.3.8. Accounting and System Shutdowns

In the case that the client for some reason fails to send a RADIUS *AccountingRequest* **STOP** packet, the accounting server will never be able to update its user statistics, but will most likely believe that the session is still active. 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, the advanced setting *Logout at shutdown* 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.

3.3.9. Limitations with NAT

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 example, when several users are behind the same network using NAT to allow network access through a single external IP address. This means that as soon as one user is authenticated, traffic coming through that NAT gateway IP address could be assumed to be coming from that one authenticated user even though it may come from other users on the same network. CorePlus RADIUS Accounting will therefore gather statistics for all the users on the network together as though they were one user instead of individuals.

3.3.10. RADIUS Advanced Settings

The following advanced settings are available with RADIUS accounting:

Allow on error

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 setting 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*

Logout at shutdown

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 been 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*

Maximum Radius Contexts

The maximum number of contexts allowed with RADIUS. This applies to RADIUS use with both accounting and authentication.

Default: *1024*

Example 3.15. RADIUS Accounting Server Setup

This example shows configuring of a local RADIUS server known as *radius-accounting* with IP address *123.04.03.01* using port *1813*.

Web Interface

1. Go to **User Authentication > Accounting Servers > Add > Radius Server**
2. Now enter:
   - **Name:** radius-accounting
   - **IP Address:** 123.04.03.01
   - **Port:** 1813
   - **Retry Timeout:** 2
   - **Shared Secret:***enter a password*
   - **Confirm Secret:***re-enter the password*
   - **Routing Table:** main
3. Click **OK**

3.4. Monitoring

The real-time performance of CorePlus can be monitored in a number of ways which are described in this section. They are:

- Using the Clavister PinPoint™ product.
- CorePlus Real-time monitor alerts.
- The CorePlus Link Monitor.
- Monitoring through an SNMP client.
- Hardware Monitoring.

3.4.1. Monitoring with PinPoint™

PinPoint is a standalone software product specifically designed for real-time monitoring of one or more Clavister Security Gateways. It is a complement to the existing CorePlus monitoring tools.

PinPoint is a visual tool where the administrator can create *dashboards* which are made up of collections of different graphical monitoring objects. The objects might be in the form of graphs or gauges or might also be in the form of meters. Each object is assigned different CorePlus parameters to monitor, for instance a gauge might indicate overall throughout. PinPoint's emphasis is on providing immediate visual feedback of what different CorePlus modules and functions are doing at any point in time and to provide the administrator with a visual representation of overall system performance.

PinPoint is free to use and does not require additional licensing. It can be downloaded from the Clavister Customer Web and the user manual is included with the product as well as being available from the Clavister website.

Choosing the right PinPoint version
Note that with WebUI enabled versions of CorePlus, the older PinPoint 1.0 version is not compatible. A later version, 1.1 or higher, is required.

3.4.2. Real-time Monitor Alerts

A number of CorePlus statistical values can graphically monitored through the Clavister PinPoint™ product as described previously. *Real-time Monitor Alert* thresholds can be specified in CorePlus for any of these monitored values so that they can have a maximum and/or a minimum numerical threshold.

Should a specified maximum or minimum threshold be crossed, CorePlus will automatically generate a log message which will be sent to all configured log receivers. All such log messages belong to the **REALTIMEMONITOR** message category which has the identity number **54**. The log message identity will therefore take the form **054XXXXX** where **XXXXX** represents the position of the statistic generating the event in the list of alert rules. A log message with identity **05400003**, for example, identifies the third rule in the rule list.

Each Monitor Alert Rule consists of the following fields:

Name	User assigned name for the rule.
Sample time	The interval in seconds between checking the statistic.
Low threshold	The lower threshold (if specified).
High threshold	A higher threshold (if specified).

Continuous This determines if an event is also generated when the threshold is crossed in the other direction. In other words, the statistic moves back to within acceptable limits. This field can be **Yes** or **No**.

Backoff The minimum number of seconds between consecutive Monitor Threshold Rule log messages. This value can be useful in preventing a flood of log messages when a statistic is repeatedly passing a threshold and then receding from it again.

3.4.3. The Link Monitor

The *Link Monitor* is a CorePlus feature that allows monitoring of one or more IP addresses external to the Clavister Security Gateway through routine ICMP "Ping" requests. If sufficient replies are not received to this polling, CorePlus makes the assumption that the common link to those IP address is down and can then initiate one of 3 configurable actions:

1. A CorePlus reconfigure.
2. A High Availability (HA) cluster failover.
3. An HA cluster failover followed by a CorePlus reconfigure.

The Link Monitor is useful in two distinct scenarios:

- An external device develops an occasional problem with its link to the Clavister Security Gateway and the physical link needs to be renegotiated. Such problems can occur sometimes with some older equipment such as ADSL Modems. For this scenario action **1. Reconfigure** should be selected.

  A reconfigure means that the CorePlus configuration will be reloaded. All connections and states are saved for the reconfiguration but reloading means all traffic is suspended for a short period and all interface links to external devices are renegotiated.

- In an HA cluster setup, the link from the master to the external Internet (or other part of a network) can be continually monitored so that should the link fail, the slave will take over (assuming that the slave has a different physical connection to the monitored address). The action chosen for HA should be either **2. Failover** or **3. Failover and reconfigure**.

  If the first action option **1. Reconfigure** is chosen in an HA cluster, then the reconfigure will also cause a failover since it will temporarily suspend the master's operation while the reconfigure takes place and the slave will take over when it detects this inactivity. If reconfiguration with failover is desirable it is better to select the option **3. Failover and reconfigure** since this does the failover first and is almost instantaneous with almost no traffic interruption. Reconfigure first is slower and results in some traffic interruption.

  To preserve all tunnels in a VPN scenario, it is best to choose the **2. Failover** option since a reconfiguration can cause some tunnels to be lost.

Link Monitoring with HA Clusters

The most common use for link monitoring is in the HA cluster scenario described above. It is important that the master and slave do not duplicate the same condition that triggered the Link Monitor. For example, if a particular router connected to the master Clavister Security Gateway was being "pinged" by Link Monitoring, the slave should not also be connected to that router. If it is, the continued triggering of a reconfiguration by the Link Monitor will then cause the slave to failover back to the master, which will then failover back to the slave again and so on.

Link Monitoring Parameters

The Link Monitor takes the following parameters:

Action Specifies which of the 3 actions described above CorePlus should take.

Addresses Specifies a group of hosts to monitor. If at least half of them do not respond, CorePlus assumes that there is a link problem. A host's responses are ignored until CorePlus has been able to reach it at least once. This means that an unreachable host can be responsible for triggering an action once but not twice.

A group of three hosts where one has been unreachable since the last configuration will therefore be treated as a two-host group until the third host becomes reachable. This also means that if a link problem triggers an action and the problem is not solved, CorePlus will not attempt to repeat the same action until the problem is solved and the hosts are again reachable.

Maximum Loss A single host is considered unreachable if this number of consecutive ping responses to that host are not replied to.

Grace Period Do not allow the link monitor to trigger an action for this number of seconds after the last reconfiguration. This avoids false positives during initial link negotiation. The default value is 45 seconds.

Send from Shared IP Address This option only appears in an HA cluster and should be used if individual public IP addresses are not available to the devices in a cluster.

3.4.4. SNMP Monitoring

Overview

Simple Network Management Protocol (SNMP) is a standardized protocol for management of network devices. An SNMP compliant client can connect to a network device which supports the SNMP protocol to query and control it.

CorePlus supports SNMP version 1 and version 2. Connection can be made by any SNMP compliant clients to devices running CorePlus. However, only query operations are permitted for security reasons. Specifically, CorePlus supports the following SNMP request operations by a client:

- The *GET REQUEST* operation
- The *GET NEXT REQUEST* operation
- The *GET BULK REQUEST* operation (SNMP Version 2c only)

The CorePlus MIB

The *Management Information Base* (MIB) is a database, usually in the form of a file, which defines the parameters on a network device that an SNMP client can query or change. The MIB file for a device running CorePlus is distributed with the standard CorePlus distribution pack as a file with the name *CLAVISTER-MIB* and this should be transferred to the hard disk of the workstation that will run the SNMP client so it can be imported by the client software. When the client runs, the MIB file is accessed to inform the client of the values that can be queried on a CorePlus device.

Defining SNMP Access

SNMP access is defined through the definition of a CorePlus *Remote* object with a *Mode* value of *SNMP*. The Remote object requires the entry of:

- **Interface** - The CorePlus interface on which SNMP requests will arrive.
- **Network** - The IP address or network from which SNMP requests will come.
- **Community** - The community string which provides password security for the accesses.

The Community String

Security for SNMP Versions 1 and 2c is handled by the *Community String* which is the same as a password for SNMP access. **The Community String should be difficult to guess** and therefore be constructed in the same way that any other password, using combinations of upper and lower case letters with digits.

Enabling an IP Rule for SNMP

The advanced setting **SNMP Before Rules** in **System > Remote Management > Advanced Settings** controls if the IP rule set checks all accesses by SNMP clients. This is by default disabled and the recommendation is to always enable this setting.

The effect of enabling this setting is to add an invisible **Allow** rule at the top of the IP rule set which automatically permits accesses on port 161 from the network and on the interface specified for SNMP access. Port 161 is usually used for SNMP and CorePlus always expects SNMP traffic on that port.

Remote Access Encryption

It should be noted that SNMP Version 1 or 2c access means that the community string will be sent as plain text over a network. This is clearly insecure if a remote client is communicating over the public Internet. It is therefore advisable to have remote access take place over an encrypted VPN tunnel or similarly secure means of communication.

Preventing SNMP Overload

The advanced setting **SNMPReqLimit** restricts the number of SNMP requests allowed per second. This can help prevent attacks through SNMP overload.

Example 3.16. Enabling SNMP Monitoring

This example enables SNMP access through the internal **lan** interface from the network **mgmt-net** using the community string *Mg1RQqR*. Since the management client is on the internal network it isn't needed to implement a VPN tunnel for it.

CLI

```
Device:/> add RemoteManagement RemoteMgmtSNMP my_snmp Interface=lan
                               Network=mgmt-net SNMPGetCommunity=Mg1RQqR
```

Should it be necessary to enable **SNMPBeforeRules** (which is enabled by default) then the command is:

```
Device:/> set Settings RemoteMgmtSettings SNMPBeforeRules=Yes
```

Web Interface

1. Goto **System > Remote Management > Add > SNMP management**
2. For **Remote access type** enter:
   - **Name:** a suitable name, for example *snmp_access*
   - **Community:** Mg1RQqR
3. For **Access Filter** enter:
   - **Interface:** lan
   - **Network:** mgmt-net
4. Click **OK**

Should it be necessary to enable **SNMPBeforeRules** (which is enabled by default) then the setting can be found in **System > Remote Management > Advanced Settings**.

3.4.4.1. SNMP Advanced Settings

The following SNMP advanced settings can be found under the **Remote Management** section in the WebUI.

SNMP Before RulesLimit

Enable SNMP traffic to the security gateway regardless of configured IP Rules.

Default: *Enabled*

SNMP Request Limit

Maximum number of SNMP requests that will be processed each second by CorePlus. Should SNMP requests exceed this rate then the excess requests will be ignored by CorePlus.

Default: *100*

System Contact

The contact person for the managed node.

Default: *N/A*

System Name

The name for the managed node.

Default: *N/A*

System Location

The physical location of the node.

Default: *N/A*

Interface Description (SNMP)

What to display in the SNMP MIB-II ifDescr variables.

Default: *Name*

Interface Alias

What to display in the SNMP ifMIB ifAlias variables.

Default: *Hardware*

3.4.5. Hardware Monitoring

Certain Clavister hardware models allow the optional monitoring by CorePlus of various operational parameters such as temperatures and fan speeds. This feature is not supported on non-Clavister hardware and to determine its availability on any Clavister hardware model, the *sysmsg* CLI command can be issued. The output will be similar to the following and the presence of the letters **HWM** indicates that hardware monitoring is possible.

```
Device:/> sysmsg

2006-06-03 12:22:23 HWM:
Via 686[A/B] found at dev[0x07], fun[0x04], base[0x00006000]
```

Caution
Using the HWM command by mistake on non-Clavister hardware can cause CorePlus to stop and require a restart.

Enabling Hardware Monitoring

The **System** > **Hardware Monitoring** section of the WebUI provides the administrator with the following settings for enabling hardware monitoring when it is available:

Enable Sensors

Enable/disable all hardware monitoring functionality.

Default: *500*

Poll Interval

Polling interval for the Hardware Monitor which is the delay in milliseconds between reading of hardware monitor values. Minimum 100, Maximum 10000.

Default: *500*

Using the *hwm* Command

To get a list of all available sensors, the command *hvm -all -verbose* can be used. Some typical output from this command for the Clavister SG4200 Series is shown below (the command options have been abbreviated).

```
Device:/> hvm -a -v

4 sensors available
```

```
Poll interval time = 500ms

Name [type][number] = low_limit] current_value [high_limit (unit)
-------------------------------------------------------------------------------
SysTemp          [TEMP  ][0] =    15.000]    23.000 [   40.000 (C)
CPUTemp          [TEMP  ][1] =    20.000]    29.000 [   40.000 (C)
CPUFan           [FANRPM][0] =  6000.000]  6619.000 [ 7000.000 (RPM)
ChassisFan       [FANRPM][1] =  8000.000]  8511.000 [10000.000 (RPM)

Time to probe sensors: 0.208 milliseconds
```

Each physical attribute listed on the left is given a range within which it should operate. When the value returned after polling falls outside this range, CorePlus optionally generates a log message which is sent to the configured log servers.

Setting the Minimum and Maximum

The minimum and maximum values shown in the output from the *hvm* command can be set through the WebUI by going to **System** > **Hardware Monitoring** > **Add** and selecting the hardware parameter to monitor. The desired operating range can then be specified.

3.4.6. Memory Monitoring

The **System** > **Hardware Monitoring** section of the WebUI provides the administrator with a number of settings connected with monitoring of available memory. These are:

Memory Poll Interval

Memory polling interval which is the delay in minutes between reading of memory values. Minimum 1, Maximum 200.

Default: *15 mins*

Memory Use Percentage

True if the memory monitor uses a percentage as the unit for monitoring, *False* if megabytes are used. Applies to Alert Level, Critical Level and Warning Level.

Default: *True*

Memory Log Repetition

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 Memory Poll Interval is triggered. If False, a message is sent when a value goes from one level to another.

Default: *False*

Alert Level

Generate an Alert log message if free memory is below this value. Disable by setting to 0. Maximum value is 10,000.

Default: *0*

Critical Level

Generate a Critical log message if free memory is below this value. Disable by setting to 0. Maximum value is 10,000.

Default: *0*

Warning Level

Generate a Warning log message if free memory is below this value. Disable by setting to 0. Maximum value 10,000.

Default: *0*

3.5. Diagnostic Tools

3.5.1. Overview

In the case of a serious system problem CorePlus provides some tools to aid in identifying the cause. These are:

- Diagnostic CLI commands
- The *pcapdump* CLI command

Both of these provide output which can be sent to support staff at Clavister to help identify the circumstances that led to the problem in question.

3.5.2. Diagnostic CLI Commands

The *stat* Command

If a serious CorePlus problem is suspected then the first step should be to use the CLI command:

```
> stat
```

The **stat** command will indicate the date and time of the last system shutdown and can indicate if there has been a serious error in CorePlus operation. It should be remembered however that the buffer which *stat* uses is cleared by certain operations such as *reconfigure* and the output will not therefore show what occurred prior to buffer clearance.

The *dconsole* Command

The next step is to use the CLI command:

```
> dconsole
```

This can be abbreviated to:

```
> dcon
```

The **dconsole** command provides a list of important events in CorePlus operation and can help to establish the date, time and nature of events leading up to a serious problem occurring. The output might look similar like the following:

```
Showing diagnose entries since 2008-05-22:
2008-06-21 11:54:58                  Start (9.00.03-0:131)
2008-06-21 11:56:16                  Stop (RECONFIGURE)
2008-06-21 11:56:21                  Start (9.00.03-0:131)
2008-06-21 11:57:29                  Stop (RECONFIGURE)
2008-06-21 11:59:31                  Start (9.00.03-0:131)
2008-06-21 11:59:49                  Stop (NORMAL)
'
'
```

dconsole output above may include a dump of the system memory in the case of serious runtime errors. This will look similar like the following:

```
'
'
Reason: Exception 'DataAbort' occurred at address 0x7aaea34
```

```
Generation date/time: 2008-07-04 14:23:56 List of loaded PE-modules:
fwloader(1.07.04): BA:0x00100000, EP:0x00101028, SS:0x0, IS:0xe7000
fwcore(89.00.03-2336): BA:0x07761038, EP:0x0007c630 Register dump:
------------------------------------------------------
r0 : 0xe1a0003c, r1 : 0x07c685dc, r2 : 0x00000004, r3 : 0x50013700,
r4 : 0x06cb2d04, r5 : 0x0753a740, r6 : 0x050ce1f8, r7 : 0x00000000,
r8 : 0x0753a79c, r9 : 0x050ce1f8, r10: 0x00000000, r11: 0x0775ff34,
r12: 0x00000004, sp : 0x0775fcec, lr : 0x079de7e4 Stack dump:
5da89306 c33613f4 c330cfc5 04411507 45515a49 86619f8b c0db0a81
4e395861 cb25b796 e3108934 932766c5 4dcff9e9 711c3463 b9cd5d1e
52149961 9324dea3 d340dc25 15458610 63582ded 689a0c54 dfb43131
02c7d971 a0ebb72c bfaae832 db216923 08ba693b 95e4de97 98d121a2
'
'
```

Although **dconsole** output may be difficult to interpret by the administrator, it can be emailed to Clavister support representatives for further investigation. The **dconsole** command supersedes the **crashdump** command found in earlier versions of CorePlus.

3.5.3. The *pcapdump* Command

A valuable diagnostic tool is the ability to examine the packets that enter and leave the interfaces of a Clavister Security Gateway. For this purpose, CorePlus provides the CLI command *pcapdump* which not only allows the examination of packet streams entering and leaving interfaces but also allows the filtering of these streams according to specified criteria.

The packets that are filtered out by *pcapdump* can then be saved in a file of type *.cap* which is the defacto *libpcap* library file format standard for packet capture.

The complete syntax of the *pcapdump* command is described in the <emphasis>CLI Reference Guide</emphasis>.

A Simple Example

An example of *pcapdump* usage is the following sequence:

```
> pcapdump -size 1024 -start int
> pcapdump -stop int
> pcapdump -show
> pcapdump -write int -filename=cap_int.cap
> pcapdump -cleanup
```

Going through this line by line we have:

1. Recording is started for the *int* interface using a buffer size of 1024 Kbytes.

```
> pcapdump -size 1024 -start int
```

2. The recording is stopped for the *int* interface.

```
> pcapdump -stop int
```

3. The dump output is displayed on the console in a summarized form.

```
> pcapdump -show
```

4. The same information is written in its complete form to a file called *cap_int.cap*.

```
> pcapdump -write int -filename=cap_int.cap
```

At this point, the file *cap_int.cap* should be downloaded to the management workstation for analysis.

5. A final cleanup is performed and all memory taken is released.

```
> pcapdump -cleanup
```

Re-using Capture Files

Since the only way to delete files from the Clavister Security Gateway is through the serial console, the recommendation is to always use the same filename when using the pcapdump *-write* option. Each new write operation will then overwrite the old file.

Running on Multiple Interfaces

It is possible to have multiple *pcapdump* executions being performed at the same time. The following points describe this feature:

1. All capture from all executions goes to the same memory buffer.

   The command can be launched multiple times with different interfaces specified. In this case the packet flow for the different executions will be grouped together in different sections of the report.

   If a clearer picture of packets flowing between interfaces is required in the output then it is best to issue one *pcapdump* command with the interfaces of interest specified.

2. If no interface is specified then the capture is done on all interfaces.

3. The *-stop* option without an interface specified will halt capture on all interfaces.

4. *pcapdump* prevents capture running more than once on the same interface by detecting command duplication.

Filter Expressions

Seeing all packets passing through a particular interface often provides an excess of information to be useful. To focus on particular types of traffic the *pcapdump* command has the option to add an *filter expression* which has one of the following forms:

-eth=<macaddr> - Filter on source or destination MAC address.
-ethsrc=<macaddr> - Filter on source MAC address.
-ethdest=<macaddr> - Filter on destination MAC address.
-ip=<ipaddr> - Filter source or destination IP address.
-ipsrc=<ipaddr> - Filter on source IP address.
-ipdest=<ipaddr> - Filter on destination IP address.
-port=<portnum> - Filter on source or destination port number.
-srcport=<portnum> - Filter on source port number.
-destport=<portnum> - Filter on destination port number.
-proto=<id> - Filter on protocol where id is the decimal protocol id.
-<protocolname> - Instead of using *-proto=*, the protocol name alone can be specified and can be one of *-tcp*, *-udp* or *-icmp*.

Downloading the Output File

As shown in one of the examples above, the *-write* option of *pcapdump* can save buffered packet information to a file on the Clavister Security Gateway.

These output files are placed into the CorePlus root directory and the file name is specified in the *pcapdump* command line, usually with a filetype of *.cap*. The name of output files must follow certain rules which are described below. Files can be downloaded to the local workstation using Secure Copy (SCP) (see Section 3.1.6, “Secure Copy”). A list of all files in the CorePlus root directory can be viewed by issuing the *ls* CLI command.

The *-cleanup* option will erase the files so cleanup should only be done after file download is complete.

Output File Naming Restrictions

The name of the file used for pcapdump output must comply with the following rules:

- Excluding the filename extension, the name may not exceed 8 characters in length.
- The filename extension cannot exceed 3 characters in length.
- The filename and extension can only contain the characters A-Z, 0-9, "-" and "_".

Combining Filters

It is possible to use several of these filter expressions together in order to further refine the packets that are of interest. For example we might want to examine the packets going to a particular destination port at a particular destination IP address.

Compatibility with Wireshark

The open source tool *Wireshark* (formerly called *Ethereal*) is an extremely useful analysis tool for examining logs of captured packets. The industry standard *.pcap* file format used by *pcapdump* with its *-write* option means that it is compatible with Wireshark.

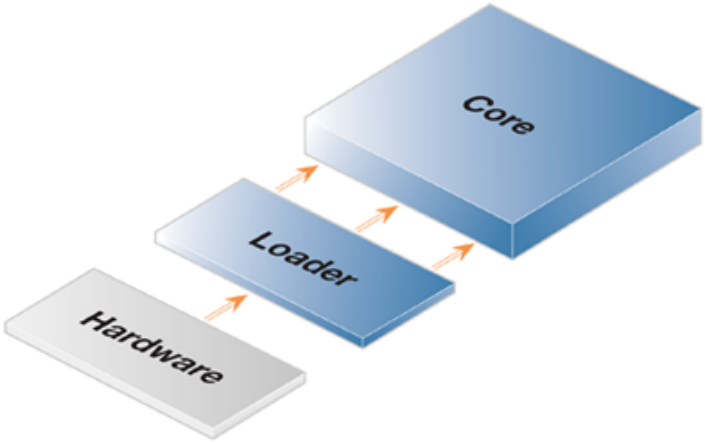
For more complete information on this topic, see *http://www.wireshark.org*.

3.6. Maintenance

3.6.1. Software Upgrades

Clavister Security Gateways are driven and controlled by CorePlus and this consists of two major components: the CorePlus *Core* and the CorePlus *Loader*. Usually it is the CorePlus Core which is the main concern of the administrator since this is the executable binary of CorePlus and contains all its principal components. The Loader, as described below, is a low level firmware loader used by the Core that is updated only with some major revisions of CorePlus.

Figure 3.1. CorePlus Firmware Structure



3.6.1.1. The *Core* Executable

Whenever new functionality is added to CorePlus, or when defects have been found and corrected, a new *Core* is produced which is the CorePlus executable binary. The new Core is packaged as a single file which is digitally signed and made available for download from the Clavister Customer Web. The Customer Web can be found at the URL: *https://clientweb.clavister.com*.

In the distribution of software upgrades, a Core file name will indicate that it is either a **Full** version designed for most hardware models or the file name will indicate a specific hardware model. If there is a Core file with a filename related to your hardware then use that file for upgrading, otherwise use the **Full** version.

Important

*The Core files for some Clavister hardware models, for example the SG10 and SG50, are specific to these hardware models and such a model specific file will have a filename which includes the model number. Do **not** use the **Full** version for upgrading the SG10 and SG50 but instead use the model specific files.*

There are two types of CorePlus Core upgrades:

- Minor upgrades, for example, from version 8.90.01 to 8.90.02, include bug fixes and minor feature enhancements and are freely available to all customers who are licensed to run the major version.
- Major upgrades, for example, from version 8.80 to 8.90, are only available for customers that have hold a valid software subscription service contract.

Checking the *Dynamic High Buffers* Setting

Make sure that the advanced setting *Dynamic High Buffers* is enabled after an upgrade. This setting ensures that sufficient memory is allocated for handling connections.

This setting requires a hardware restart for a new value to take effect. A reconfiguration is not sufficient.

Where CorePlus is handling tens of thousands of simultaneous connections then it may be necessary to manually set a value above the automatic value. If the **HighBuffers** value is set too high this may impact throughput latency.

3.6.2. 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 7.5, “Intrusion Detection and Prevention”
- Section 7.4, “Anti-Virus Scanning”
- Section 7.3, “Web Content Filtering”
- Appendix A, *Subscribing to Security Updates*

3.6.3. Creating Backup Files

The administrator has the ability to take a snapshot of a CorePlus system at a given point in time and restore it when necessary. The snapshot can be of two types:

- A *configuration backup* which does not include the installed CorePlus version. This is useful if the CorePlus version does not change.
- A *system backup* which is a complete backup of both the configuration and the installed CorePlus software. This is useful if both the configuration is to be changed and the CorePlus version upgraded.

Backup files can be created both by downloading the files directly from the Clavister Security Gateway using SCP (Secure Copy) or alternatively using the WebUI. It cannot be done though the CLI.

Operation Interruption

Backups can be created at any time without disturbing CorePlus operation. After restoring a backup it is necessary to perform an **Activate** to make the restored configuration/system active.

Restoring and activating a configuration-only backup should not, in most cases, disturb system operation. Complete system restore, however, is more involved and will require that CorePlus reinitializes, with the loss of all existing connections. Initialization may require some seconds to complete depending on the hardware type and normal operation will not be possible during this time.

Backup and Restore using SCP

There are two files located in the CorePlus root directory:

- *config.bak* - This is the backup of the current configuration.
- *full.bak* - This is the backup of the complete system.

SCP can be used to download either of these files. When the download is complete the filename will be altered to include the date. For example, *full.bak* might become *full-20081121.bak* to show it is a snapshot of the state on November 21st, 2008.

To restore a backup file, the administrator should upload the file to the Clavister Security Gateway. The name of the file does not need to be changed in any way and can retain the date since CorePlus will read a header in the file to determine what it is.

Backup and Restore using the WebUI

As an alternative to using SCP, the administrator can initiate a backup or restore of the configuration or complete system directly through the WebUI. The examples below illustrates how this is done.

Example 3.17. Backing up the entire system.

In this example we will backup the entire system on 12 December 2008.

Web Interface

1. Go to **Maintenance > Backup**
2. The *Backup* dialog will be shown
3. Press the **Backup configuration** button
4. A file dialog is shown - choose a directory for the created file
5. Download of the backup file will then start

Note: Backups do not contain everything

Backups include only static information from the CorePlus configuration. Dynamic information such as the DHCP server lease database or Anti-Virus/IDP databases will not be backed up.

3.6.4. Restore to Factory Defaults

A *restore to factory defaults* can be applied so that it is possible to return to the original hardware state that existed when the Clavister Security Gateway was shipped by Clavister. When a restore is applied, data such as the IDP and Ant-Virus databases as well as *pcapdump* files are not deleted. These must be explicitly removed by using the appropriate command.

Example 3.18. Complete Hardware Reset to Factory Defaults

CLI

```
Device:/> reset -unit
```

Web Interface

1. Go to **Maintenance > Reset**
2. Select **Restore the entire unit to factory defaults** then confirm and wait for the restore to complete.

Warning
DO NOT ABORT THE RESET TO FACTORY DEFAULTS PROCESS. *If aborted the Clavister Security Gateway can cease to function properly.*

End of Life Procedures

The restore to factory defaults option should also be used as part of the end of life procedure when a Clavister Security Gateway is taken out of operation and will no longer be used. As part of the decommissioning procedure, a restore to factory defaults should always be run in order to remove all sensitive information such as VPN settings.

As a further precaution at the end of the product's life, it also recommended that the memory media in a Clavister Security Gateway is destroyed and certified as destroyed by a suitable provider of computer disposal services.

3.6.5. Listing and Adding Ethernet Ports

The CLI command *pciscan* is a tool that is used in listing and upgrading the Ethernet ports for the Clavister Security Gateway. It is usually relevant only for non-Clavister hardware although it can be used for examining the PCI hardware in Clavister devices.

Checking Ethernet Ports

If the *pciscan* command is used without parameters then a list of all the supported Ethernet ports in the Clavister Security Gateway is obtained:

```
Device:/> pciscan

Idx VendorID DeviceID Bus  Slot IRQ Driver Info
012 0x8086   0x1010    2    4   0 "E1000"  Intel NETWORK - ETHERNET
013 0x8086   0x1010    2    4   1 "E1000"  Intel NETWORK - ETHERNET
020 0x8086   0x1012    5    4   0 "E1000"  Intel NETWORK - ETHERNET
021 0x8086   0x1012    5    4   1 "E1000"  Intel NETWORK - ETHERNET
022 0x8086   0x1012    6    4   0 "E1000"  Intel NETWORK - ETHERNET
"
"
```

Each line in the output shows the current configuration for each Ethernet port.

To see all Ethernet ports on the Clavister Security Gateway, regardless if they are supported or not, the command is:

```
Device:/> pciscan -ethernet
```

To see all hardware on the PCI bus, both Ethernet and non-Ethernet, the command is:

```
Device:/> pciscan -all
```

Automatic Reconfiguration

If a new Ethernet port is added to the Clavister Security Gateway, this can be detected and the configuration updated automatically with the command:

```
Device:/> pciscan -cfgupdate
```

Forcing the Choice of Driver

When using non-Clavister hardware, there may be PCI cards installed that require a driver that cannot be automatically selected using the *-cfgupdate* option. In such a case the administrator can explicitly choose the driver from a list using the *-force_driver* option.

The index number of the PCI card is first identified from the output of the *pciscan -all* command. If the card number index is found to be *7* and the driver to choose is *E1000* then the command is:

```
Device:/> pciscan -force_driver 7 E1000
```

The command offers tab completion for the available driver names so it is not necessary to know them in advance. It may be the case that a process of trial and error is required to identify the correct driver for the card.

3.7. Licensing

3.7.1. Overview

To use CorePlus in a live environment, a CorePlus *license file* must be deployed to the Clavister Security Gateway. A unique license file is needed for each processor on which CorePlus runs and in the case of the multi-bladed systems, a different license file is required for each installed blade since these operate as autonomous Clavister Security Gateways.

The purpose of the license file is to define what the capabilities and limitations a CorePlus installation will have. Such capabilities include parameters such as the number of VPN tunnels allowed, the number of routing tables or the number of virtual interfaces. This manual does not cover license pricing Please refer to your local Clavister sales office for pricing information.

3.7.2. Evaluation Mode

Each installation of CorePlus requires a unique license, issued by Clavister, to have full functionality. Without correct licensing, CorePlus will only operate for 2 hours from hardware boot up. After 2 hours, CorePlus will cease to function and will output an expiry message to the local console.

The hardware must be re-booted to enable CorePlus for a further 2 hour period although there are no limits on how many times this can be done. The 2 hour limit is designed to allow a reasonable window of time for product evaluation. To remove the time limit, a valid license must be made available to the CorePlus by *binding* it to the installation.

3.7.3. License Purchase

A license is required for each of the two types of CorePlus usage:

1. A software-only license purchase is made and a License *Registration Key* is supplied to the licensee by Clavister. Delivery of the software-only registration key can be in one of 2 ways:

   - Software can be downloaded from the Clavister website and a registration key from a *Certificate of Authenticity* is then sent to the customer via email after purchase.

   - If preferred, a CD-ROM can be sent to the customer containing all software, along with the *Certificate of Authenticity* that bears a registration key. This can also be done in addition to email code notification if the customer requires it.

2. When purchasing Clavister hardware, the registration key is supplied in the following ways for different hardware:

   - For the SG10, SG50 and SG3100 units, the registration key is written on a label on the underside of the hardware unit. The MAC address of one of the interfaces is also written on this label.

   - For the SG4200 and SG4400 a *Certificate of Authenticity* bearing the registration key is packaged with the hardware.

   - For the SG5500, a separate Certificate of Authenticity is packaged with each Secure Blade Module (SBM) purchased since each blade is an autonomous processor running its own copy of CorePlus and each requires a separate license.

It should be noted that the registration key and the license number are different. The registration key may not be required again after initial registration but should be retained as proof of purchase. The license number has the form of 4 sets of 4 digits of the form: nnnn-nnnn-nnnn-nnnn.

HA Cluster Licensing

In a High Availability Cluster, two identical licenses must be purchased, one for the master and one for the slave unit. Both licenses must include the ability to allow HA clustering.

Important

In any installation with more than one Clavister Security Gateway it is important to always use the right license file for the right Clavister Security Gateway. If licenses are not matched correctly to the hardware, complex administrative problems can arise later which can cause delays in rectifying a hardware fault.

3.7.4. The Clavister Customer Web

The Clavister Customer Web is for the administration of CorePlus licenses. It can be used for first time license registration as well as allowing customers to download license updates and to get access to software upgrades.

Customer Web License Registration

Following purchase, the administrator can access the Clavister Customer Web to complete the registration process and obtain the corresponding license file. The first time it is accessed, the Customer Web requires that you enter your contact details, the registration key, as well as the MAC address of one of the Ethernet ports of the hardware on which the license will be used. A user name and password must also be specified for subsequent Customer Web access. After successful registration a license file can then be downloaded.

3.7.5. License Files

A license file is a text file that defines all the capabilities allowed by the license plus a digital signature to ensure the file cannot be altered. It can be opened and read in a normal text editor and the file name is always the license number followed by a file type of *.lic*. Once the license file is available to a CorePlus installation, the 2 hour usage time limit disappears and functionality will be limited only by the usage limits specified by the license file, for example the number of VPN tunnels.

3.7.6. Registering Additional Licenses

A licensee need register themselves only once on the Clavister Customer Web. After logging in they are shown a list of existing licenses and have the option to register additional licenses. Additional license registration requires only the new registration key with the associated MAC address. A license file for the new license can then be downloaded.

Changing MAC Addresses

The Customer Web also allows the option to change the MAC Address associated with a given license. This can only be done twice per license.

3.7.7. Lockdown Mode

CorePlus will enter a state known as *Lockdown Mode* if certain license violations occur. While in Lockdown Mode, only remote management traffic is allowed by the Clavister Security Gateway and all other traffic will be dropped. Unlike the two hour Demonstration Mode, there is no time limit on Lockdown Mode.

Lockdown Mode is usually caused by the license file bound to the Clavister Security Gateway being in some way invalid. Conditions that trigger Lockdown Mode include license date expiry, invalid MAC address or invalid license file signature. It is also triggered if the administrator tries to configure CorePlus beyond the limits of its license file. Trying to configure more VPN tunnels than the license allows is an example of this.

When Lockdown Mode in entered, the condition can be terminated by binding a valid License to CorePlus or removing the configuration violation that triggered the condition. Unbinding the license will cause it to enter the 2 hour demonstration mode from Lockdown Mode, and this may be necessary to allow traffic to flow to the Internet in order to download a new license file.

3.7.8. Uploading Licenses

When a license file needs to be uploaded to the Clavister Security Gateway, this can be done in one of two ways:

1. By using SCP (Secure Copy).

   Only one license file can exist on the Clavister Security Gateway. The name of the file is not mandatory, nor the location since CorePlus will detect the file by examining its contents. As a convention it is strongly recommended that the license file is called *license.lic* and it is uploaded to the top level of the CorePlus directory structure.

   Under Linux the SCP upload command to a security gateway called *sgw_name* might be:

   ```
   > scp license.lic user@sgw_name:license.lic
   ```

   Under windows the SCP upload would be done using an appropriate utility with SCP support.

2. Through the WebUI:

   - Select **Maintenance** from the WebUI's menu bar.
   - Choose the **Upgrade** option.
   - Select the license file to upload.
   - Click the **Upload license** button.

After license uploading is complete, a CorePlus *reconfigure* is initiated automatically which means that there will be a brief suspension of CorePlus operation. For this reason, it is recommended that license upload is done during routine maintenance times or other non-critical periods.

Should the license be invalid, CorePlus will enter *lockdown mode* as described in 3.7.7 above. Deleting the license entirely will put CorePlus into the 2 hour *evaluation mode* which is also described in 3.7.2.

Chapter 4. Fundamentals

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

4.1. The Address Book

4.1.1. 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.

4.1.2. 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 or a range of IP addresses and 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 9, *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 on. 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.

Example 4.1. Adding an IP Host

This example adds the IP host *www_srv1* with IP address *192.168.10.16* to the Address Book:

CLI

```
Device:/> add Address IP4Address www_srv1 Address=192.168.10.16
```

Web Interface

1. Go to **Objects > Address Book > Add > IP address**
2. Specify a suitable name for the IP host, in this case *wwww_srv1*
3. Enter *192.168.10.16* for the **IP Address**
4. Click **OK**

Example 4.2. Adding an IP Network

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

CLI

```
Device:/> add Address IP4Address wwwsrvnet Address=192.168.10.0/24
```

Web Interface

1. Go to **Objects > Address Book > Add > IP address**
2. Specify a suitable name for the IP network, for example *wwwsrvnet*
3. Enter *192.168.10.0/24* as the **IP Address**
4. Click **OK**

Example 4.3. 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*:

CLI

```
Device:/> add Address IP4Address wwwservers Address=192.168.10.16-192.168.10.21
```

Web Interface

1. Go to **Objects > Address Book > Add > IP address**
2. Specify a suitable name for the IP Range, for example *wwwservers*.
3. Enter *192.168.10.16-192.168.10.21* as the **IP Address**
4. Click **OK**

Example 4.4. Deleting an Address Object

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

CLI

```
Device:/> delete Address IP4Address wwwsrv1
```

Web Interface

1. Go to **Objects > Address Book**
2. Select and right-click the address object *wwwsrv1* in the grid
3. Choose **Delete** from the menu
4. Click **OK**

4.1.3. Ethernet Addresses

Ethernet Address objects are used to define symbolic names for Ethernet addresses (also known as MAC addresses). This is useful, for example, 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.

Example 4.5. Adding an Ethernet Address

The following example adds an Ethernet Address object named *wwwsrv1_mac* with a numerical MAC address of *08-a3-67-bc-2e-f2*:

CLI

```
Device:/> add Address EthernetAddress wwwsrv1_mac Address=08-a3-67-bc-2e-f2
```

Web Interface

1. Go to **Objects > Address Book > Add > Ethernet Address**
2. Specify a suitable name for the Ethernet Address object, for example *wwwsrv1_mac*
3. Enter *08-a3-67-bc-2e-f2* as the **MAC Address**
4. Click **OK**

4.1.4. 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 example *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 on. 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 cannot be combined with Ethernet addresses.

4.1.5. Auto-Generated Address Objects

To simplify the configuration, several address objects are automatically generated when the system runs 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, two IP Address objects are pre-defined; one object for the IP address of the actual interface, and one object representing the local network for that interface.

Interface IP address objects are named <*interface-name*>*_ip* and network objects are named <*interface-name*>*_net*. As an example, an interface named *lan* will have an associated interface IP object named *ip_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 an DHCP server. The object will have an empty IP address (0.0.0.0/0), and the correct IP address for the default gateway needs to be defined unless DHCP is being used.

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

4.1.6. Address Book Folders

In order to help organise large numbers of entries in the address book, it is possible to create Address Book *folders*. These folders are just like a folder in a computer's file system. They are created with a given name and can then be used to contain all the IP address objects that are related together as a group.

Using folders is simply a way for the administrator to conveniently divide up Address Book entries and no special properties are given to entries in different folders. CorePlus continues to see all entries as though they were in large table of IP address objects.

The folder concept is also used by CorePlus in IP rule sets, where related IP rules can be grouped together in administrator created folders.

4.2. Services

4.2.1. 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.

However, 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 as Objects

Services are passive objects in that they cannot carry out any action in the system on their own. Instead, Service objects are used frequently in the various security policies defined by rule sets. 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 Clavister Security Gateway. For more information on how service objects are being used with IP rules, see Section 4.5, “The IP Rule Set”.

Pre-defined Services

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

Example 4.6. Listing the Available Services

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

CLI

```
Device:/> show Service
```

The output will look similar to the following listing:

```
ServiceGroup

   Name          Comments
   ------------  --------------------------------------------------
   all_services  All ICMP, TCP and UDP services
   all_tcpudp    All TCP and UDP services
   ipsec-suite   The IPsec+IKE suite
   l2tp-ipsec    L2TP using IPsec for encryption and authentication
   l2tp-raw      L2TP control and transport, unencrypted
   pptp-suite    PPTP control and transport

ServiceICMP
...
```

Web Interface

1. Go to **Objects > Services**

Example 4.7. Viewing a Specific Service

To view a specific service in the system:

CLI

```
Device:/> show Service ServiceTCPUDP echo
```

The output will look similar to the following listing:

```
          Property  Value
 -----------------  ---------------
             Name:  echo
 DestinationPorts:  7
             Type:  TCPUDP (TCP/UDP)
      SourcePorts:  0-65535
   PassICMPReturn:  No
              ALG:  (none)
      MaxSessions:  1000
         Comments:  Echo service
```

Web Interface

1. Go to **Objects > Services**
2. Select the specific service object in the table
3. A listing all services will be presented

4.2.2. 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 overhead 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.

TCP and UDP Service Definition

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 on. For these types of Service, 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 used not just for destination ports. Source port definitions can follow the same conventions, although it is most usual that the source ports are left as the default value which is 0-65535 and this corresponds to all possible source ports.

Example 4.8. Adding a TCP/UDP Service

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

CLI

```
Device:/> add Service ServiceTCPUDP MySQL DestinationPorts=3306 Type=TCP
```

Web Interface

1. Go to **Objects > Services > Add > TCP/UDP service**
2. Specify a suitable name for the service, for example *MySQL*
3. Now enter:
   - **Type:** TCP
   - **Source:** 0-65535
   - **Destination:** 3306
4. 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:

SYN Flood Protection A TCP based service can be configured to enable protection against *SYN Flood* attacks. For more details on how this feature works see Section 7.6.8, “TCP SYN Flood Attacks”.

Passing ICMP Errors 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.

Application Layer Gateway A TCP/UDP Service can be linked to an *Application Layer Gateway* to enable deeper inspection of certain protocols. For more information see Section 7.2, “ALGs”.

Max Sessions

An important parameter associated with a Service is *Max Sessions*. This parameter is allocated a default value when the Service is associated with an ALG. The default value varies according to the

ALG it is associated with. If the default is, for example *100*, this would mean that only 100 connections are allowed in total for this Service across all interfaces.

For a Service involving, for instance an HTTP ALG, the default value can often be too low if there are large numbers of clients connecting through the Clavister Security Gateway. It is therefore recommended to consider if a higher value is required for a particular scenario.

4.2.3. 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 are 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.

4.2.4. 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 using the concept of *Custom IP Protocol Services*. 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 CorePlus 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 4.9. Adding an IP Protocol Service

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

CLI

```
Device:/> add Service ServiceIPProto VRRP IPProto=112
```

Web Interface

1. Go to **Objects > Services > Add > IP protocol service**
2. Specify a suitable name for the service, for example *VRRP*
3. Enter *112* in the **IP Protocol** control
4. Optionally enter *Virtual Router Redundancy Protocol* in the **Comments** control
5. Click **OK**

4.3. Interfaces

4.3.1. 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, see Section 4.3.2, “Ethernet Interfaces”.

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 4.3.3, “VLAN”.
- *PPPoE* (PPP-over-Ethernet) interfaces for connections to PPPoE servers. For more information about PPPoE, please see Section 4.3.4, “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. For example, when routing traffic over an IPsec interface, 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 10.3, “IPsec Components”.
- *PPTP/L2TP* interfaces are used as end-points for PPTP or L2TP tunnels. For more information about PPTP/L2TP, please see Section 10.5, “PPTP/L2TP”.

- *GRE* interfaces are used to establish GRE tunnels. For more information about GRE, please see Section 4.3.5, “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 almost exclusively used for *Virtual Routing* scenarios.

For more information about loopback interfaces, please see Section 4.3.6, “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 subsytems 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 which are named **any** and **core**. The meaning of these are:

- **any** represents all possible interfaces including the **core** interface.
- **core** indicates that it is CorePlus itself that will deal with traffic to and from this interface. Examples of the use of **core** are when the Clavister Security Gateway acts as a PPTP or L2TP server or responds 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.

Disabling an Interface

Should it be desirable to disable an interface so that no traffic can flow through it, this can be done with the CLI using the command:

```
Device:/> set Interface Ethernet <interface-name> -disable
```

Where <*interface-name*> is the interface to be disabled. To re-enable an interface, the command is:

```
Device:/> set Interface Ethernet <interface-name> -enable
```

4.3.2. Ethernet Interfaces

CorePlus supports 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 Ethernet to Fast Ethernet 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 dependent on the hardware model. A smaller Clavister model will have a limited number of Fast Ethernet ports as integrated product components, while a more powerful unit designed for telecom applications might be expandable with separate port modules.

If CorePlus is being run on non-Clavister Intel x86 compatible server hardware, it is likely that the Ethernet hardware ports are implemented using common Ethernet PCI adapters.

Note

Some systems use an integrated layer 2 switch for providing additional physical Ethernet ports. Such additional ports are seen as a single interface by CorePlus.

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 on.

The names of the Ethernet interfaces can be changed to better reflect their usage. For example, 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: Interface enumeration

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.

Ethernet 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 4.1.5, “Auto-Generated Address Objects”. When the system is first started, all unconfigured Ethernet interfaces will be assigned default addresses from the *localhost* subnet (*127.0.0.0/8*).

Tip: Specifying multiple IP addresses on an interface

Multiple IP addresses can be specified for an Ethernet interface by using the ARP Publish feature. (For more information, see Section 4.4, “ARP”).

Changing the IP Address of an Ethernet Interface

To change the IP address on an interface, we can use one of two methods:

- Change the IP address directly on the interface. For instance, if we want to change the IP address of the **lan** interface to *10.1.1.2*, we could use the CLI command:

```
Device:/> set Interface Ethernet lan IP=10.1.1.2
```

  As explained next, this way of changing the IP address is not recommended.

- Instead, the **ip_lan** object in the CorePlus *Address Book* should be assigned the new address since it is this object that is used by many other CorePlus objects such as IP rules. The CLI command to do this would be:

```
Device:/> set Address IP4Address ip_lan Address=10.1.1.2
```

  This same operation could also be done through the Web Interface.

A summary of CLI commands that can be used with Ethernet interfaces can be found in Section 4.3.2.1, “Useful CLI Commands for Ethernet Interfaces”.

Network Addresses

In addition to the interface IP address, a *Network* address is also specified for an Ethernet interface. The Network address provides information to CorePlus about what IP addresses are directly reachable through the interface. In other words, those residing on the same LAN segment as the interface itself. In the routing table associated with the interface, CorePlus will automatically create a direct route to the specified network over the actual interface.

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 behavior.

The all-nets IP address object

*The **all-nets** IP 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 on this topic, see Section 5.6, “Multicast Routing”.*

Specifying a Default Gateway

A *Default Gateway* address can optionally be specified for an Ethernet interface. This is a normally the address of a router and very often the router which acts as the gateway to the Internet.

When a default gateway is specified, a route to the network *all-nets* needs to be be created in the CorePlus routing table for the interface. This means that any traffic which does not have a more specific matching route will be sent via that interface to the default gateway. The creation of a default route can be performed automatically by enabling an advanced option for an interface.

Normally, only one default *all-nets* route to the default gateway needs to exist in the routing table.

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 4.1.5, “Auto-Generated Address Objects”. However, you can specify your own set of address objects if you wish to modify this behavior.

Example 4.10. Enabling DHCP

CLI

```
Device:/> set Interface Ethernet wan DHCPEnabled=Yes
```

Web Interface

1. Go to **Interfaces > Ethernet**
2. In the grid, click on the ethernet object of interest
3. Enable the **Enable DHCP client** option
4. Click **OK**

4.3.2.1. Useful CLI Commands for Ethernet Interfaces

This section summarizes the CLI commands most commonly used for examining and manipulating CorePlus Ethernet interfaces.

Ethernet interfaces can also be examined through the Web Interface but for some operations the CLI must be used.

To show the current interface assigned to the IP address *wan_ip*:

```
Device:/> show Address IP4Address InterfaceAddresses/wan_ip

             Property  Value
--------------------  --------------------------
                 Name:  wan_ip
              Address:  0.0.0.0
       UserAuthGroups:  <empty>
 NoDefinedCredentials:  No
             Comments:  IP address of interface wan
```

To show the current interface assigned to the network *wan_net*:

```
Device:/> show Address IP4Address InterfaceAddresses/wan_net

             Property  Value
--------------------  ------------------------
                Name:  wan_net
             Address:  0.0.0.0/0
      UserAuthGroups:  <empty>
NoDefinedCredentials:  No
            Comments:  Network on interface wan
```

To show the current interface assigned to the gateway *wan_gw*:

```
Device:/> show Address IP4Address InterfaceAddresses/wan_gw

             Property  Value
--------------------  ---------------------------------
                 Name:  wan_gw
              Address:  0.0.0.0
      UserAuthGroups:  <empty>
NoDefinedCredentials:  No
             Comments:  Default gateway for interface wan
```

By using the tab key at the end of a line, tab completion can be used to complete the command:

```
Device:/> show Address IP4Address InterfaceAddresses/wan_<tab>

[<Category>] [<Type> [<Identifier>]]:

InterfaceAddresses/wan_br    InterfaceAddresses/wan_gw
InterfaceAddresses/wan_dns1  InterfaceAddresses/wan_ip
InterfaceAddresses/wan_dns2  InterfaceAddresses/wan_net
```

Here, tab completion is used again at the end of the command line:

```
Device:/> set Address IP4Address<tab>

[<Category>] <Type> [<Identifier>]:

dnsserver1_ip              InterfaceAddresses/wan_br timesyncsrv1_ip
InterfaceAddresses/aux_ip  InterfaceAddresses/wan_dns1
InterfaceAddresses/aux_net InterfaceAddresses/wan_dns2
InterfaceAddresses/dmz_ip  InterfaceAddresses/wan_gw
InterfaceAddresses/dmz_net InterfaceAddresses/wan_ip
InterfaceAddresses/lan_ip  InterfaceAddresses/wan_net
InterfaceAddresses/lan_net Server
```

The CLI can be used to set the address of the interface:

```
Device:/> set Address IP4Address
                InterfaceAddresses/wan_ip Address=172.16.5.1

Modified IP4Address InterfaceAddresses/wan_ip.
```

The CLI can be used to enable DHCP on the interface:

```
Device:/> set Interface Ethernet wan DHCPEnabled=yes

Modified Ethernet wan.
```

Some interface settings are accessible only through a related set of CLI commands. These are particularly useful if Clavister hardware has been replaced and Ethernet card settings are to be changed, or if configuring the interfaces when running CorePlus on non-Clavister hardware. For example, to display Ethernet port information use the command:

```
Device:/> show EthernetDevice
```

This command shows all Ethernet interfaces defined. This list includes those interfaces deleted but before an *activate* has been done. Deletions will be indicated with a "-" symbol before their name. If a deleted interface in the list is to be restored, this can be done with the *undelete* command:

```
Device:/> undelete EthernetDevice <interface>
```

The following command can also be used to list interface information:

```
Device:/> show Ethernet Interface
```

The *set* command can be used to control an Ethernet interface. For example, to enable an interface lan we can use the command:

```
Device:/> set EthernetDevice lan -enable
```

To set the driver on an Ethernet interface card the command is:

```
Device:/> set EthernetDevice lan EthernetDriver=<driver>
                        PCIBus=<X> PCISlot=<Y> PCIPort=<Z>
```

For example, if the driver name is *IXP4NPEEthernetDriver* for the bus, slot, port combination *0, 0, 2* on the *wan* interface, the *set* command would be:

```
Device:/> set EthernetDevice lan EthernetDriver=IXP4NPEEthernetDriver
                        PCIBus=0 PCISlot=0 PCIPort=2
```

For a complete list of all CLI options see the *CLI Reference Guide*.

4.3.2.2. Advanced Ethernet Settings

This section details the advanced settings available for CorePlus Ethernet interfaces.

DHCP Settings

Below is a list of the advanced DHCP settings for CorePlus Ethernet interfaces.

DHCP_AllowGlobalBcast

Allow DHCP server to assign 255.255.255.255 as broadcast. (Non-standard.)

Default: *Disabled*

DHCP_DisableArpOnOffer

Disable the ARP check done by CorePlus on the offered IP. The check issues an ARP request to see if the IP address is already in use.

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_ValidateBcast

Require that the assigned broadcast address is the highest address in the assigned network.

Default: *Enabled*

DHCP_MinimumLeaseTime

Minimum lease time (seconds) accepted from the DHCP server.

Default: *60*

Hardware Settings

Below is a list of the advanced hardware settings that are available for CorePlus Ethernet interfaces. These settings are only relevant to CorePlus running on non-Clavister hardware.

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*

Ringsize_yukon_rx

Size of Yukon receive ring (per interface).

Default: *256*

Ringsize_yukon_tx

Size of Yukon send ring (per interface).

Default: *256*

Ringsize_yukonii_rx

Size of Yukon-II receive ring (per interface).

Default: *256*

Ringsize_yukonii_tx

Size of Yukon-II send ring (per interface).

Default: *256*

Interface Monitor Settings

Below is a list of the monitor settings that are available for CorePlus Ethernet interfaces.

IfaceMon_e1000

Enable interface monitor for e1000 interfaces.

Default: *Enabled*

IfaceMon_BelowCPULoad

Temporarily disable ifacemon if CPU load goes above this percentage.

Default: *80*

IfaceMon_BelowIfaceLoad

Temporarily disable ifacemon on and interface if network load on the interface goes above this percentage.

Default: *70*

IfaceMon_ErrorTime

How long a problem must persist before an interface is reset.

Default: *10*

IfaceMon_MinInterval

Minimum interval between two resets of the same interface.

Default: *30*

IfaceMon_RxErrorPerc

Percentage of errors in received packets at which to declare a problem.

Default: *20*

IfaceMon_TxErrorPerc

Percentage of errors in sent packets at which to declare a problem.

Default: *7*

4.3.3. VLAN

Overview

Virtual LANs (VLANs) are useful in several different scenarios, for instance, when filtering of traffic is needed between different VLANs in an organization, or for any other reason where the administrator would like to expand the number of interfaces.

Virtual LAN support in CorePlus allows the definition of one or more *Virtual LAN interfaces* to be associated with a particular physical interface. These are then considered to be logical interfaces by CorePlus and can be treated like physical interfaces in rule sets and routing tables.

VLAN Operation

CorePlus follows the IEEE 802.1Q specification for VLAN. On a protocol level, VLAN works by adding a *Virtual LAN Identifier* (VLAN ID) to Ethernet frame headers. The VLAN ID is a number from 0 up to 4095 which is used to identify the specific Virtual LAN to which the frame belongs. In this way, Ethernet frames can belong to different Virtual LANs, but can still share the same physical interface. With CorePlus, the VLAN ID must be unique for the physical interface and the same VLAN ID can be used on different physical interfaces.

Packets received through Ethernet frames on a physical interface by CorePlus, are examined for a VLAN ID. If a VLAN ID is found and a matching VLAN interface has been defined for that interface, CorePlus will use the VLAN interface as the source interface in further processing with rule sets.

If there is no VLAN ID attached to an Ethernet frame received on the physical interface then the frame is treated as being received on the physical interface and not on any VLAN interface that may be defined.

License Limitations

The number of VLAN interfaces that can be defined for a CorePlus installation is limited by the parameters of the license used. Some licenses may restrict the total number of VLANs allowed in a CorePlus installation. License upgrades can be purchased to increase this limit if required.

Summary of VLAN Setup

It is important to understand that the administrator should treat a VLAN interface just like a physical interface in that they require at least IP rules and routes to be defined in order to function. If, for instance, no **Allow** rule is defined in the IP rule set for a VLAN interface then packets arriving on that interface will be dropped. Below are the key steps for setting up a VLAN interface.

1. Assign a name to the VLAN interface.
2. Select the physical interface for the VLAN.
3. Assign a **VLAN ID** that is unique on the physical interface.
4. Optionally specify an IP address for the VLAN.
5. Optionally specify an IP broadcast address for the VLAN.
6. Create the required route(s) for the VLAN in the appropriate routing table.
7. Create rules in the IP rule set to allow traffic through on the VLAN interface.

Example 4.11. Defining a VLAN

This simple example defines a virtual LAN called *VLAN10* with 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.

CLI

```
Device:/> add Interface VLAN VLAN10 Ethernet=lan Network=all-nets VLANID=10
```

Web Interface

1. Go to **Interfaces > VLAN > Add > VLAN**
2. Enter a suitable name for the VLAN, in this case *VLAN10*
3. Now enter:
   - **Interface:** lan
   - **VLAN ID:** 10
4. Click **OK**

VLAN advanced settings

There is a single advanced setting for VLAN:

Unknown VLAN Tags

What to do with VLAN packets tagged with an unknown ID.

Default: *DropLog*

4.3.4. PPPoE

4.3.4.1. Overview

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 ISP 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

The PPP Protocol

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.

PPP Authentication

PPP authentication is optional with PPP. Authentication protocols supported are *Password Authentication Protocol* (PAP), *Challenge Handshake Authentication Protocol* (CHAP) and *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.

4.3.4.2. PPPoE Client Configuration

The PPPoE interface

Since the PPPoE protocol runs PPP over Ethernet, the security 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 security 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 security 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 4.12. Configuring a PPPoE client

This example shows how to configure a PPPoE client on the *wan* interface with traffic routed over PPPoE.

CLI

```
Device:/> add Interface PPPoETunnel PPPoEClient EthernetInterface=wan
     Network=all-nets Username=exampleuser Password=examplepw
```

Web Interface

1. Go to **Interfaces > PPPoE > Add > PPoE Tunnel**
2. Then enter:
   - **Name:** PPPoEClient
   - **Physical Interface:** wan
   - **Remote Network:** all-nets (as we will route all traffic into the tunnel)
   - **Service Name:** Service name provided by the service provider
   - **Username:** Username provided by the service provider
   - **Password:** Password provided by the service provider
   - **Confirm Password:** Retype the password
   - Under **Authentication** specify which authentication protocol to use (the default settings will be used if not specified)
   - Disable the option **Enable dial-on-demand**
   - Under **Advanced**, if **Add route for remote network** is enabled then a new route will be added for the interface
3. Click **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.

4.3.5. GRE Tunnels

Overview

The *Generic Router Encapsulation* (GRE) protocol is a simple, encapsulating protocol that can be used whenever there is a need to tunnel traffic across networks and/or through network devices. GRE does not provide any security features but this means that its use has extremely low overhead.

Using GRE

GRE is typically used to provide a method of connecting two networks together across a third network such as the Internet. The two networks being connected together communicate with a common protocol which is tunneled using GRE through the intervening network. Examples of GRE usage are:

- Traversing network equipment that blocks a particular protocol.
- Tunneling IPv6 traffic across an IPv4 network.
- Where a UDP data stream is to be multicast and it is necessary to transit through a network device which does not support multicasting. GRE allows tunneling though the network device.

GRE Security and Performance

A GRE tunnel does not use any encryption for the communication and is therefore not, in itself,

secure. Any security must come from the protocol being tunneled. The advantage of GRE's lack of encryption is the high performance which is achievable because of the low traffic processing overhead. The lack of encryption can be acceptable in some circumstances if the tunneling is done across an internal network that is not public.

Setting Up GRE

Like other tunnels in CorePlus such as an IPsec tunnel, a GRE Tunnel is treated as a logical interface by CorePlus, with the same filtering, traffic shaping and configuration capabilities as a standard interface. The GRE options are:

- **IP Address** - This is the IP address of the sending interface. This is optional and can be left blank. If it is left blank then the sending IP address will default to the local host address of *127.0.0.1*.
- **Remote Network** - The remote network which the GRE tunnel will connect with.
- **Remote Endpoint** - This is the IP address of the remote device which the tunnel will connect with.
- **Outer PBR Table** - This defines the routing table to be used for the tunnel itself and not the traffic that it is carrying. In other words: the table used to look up the tunnel endpoint.
- **Use Session Key** - A unique number can optionally be specified for the tunnel. This allows more than one GRE tunnel to run between the same two endpoints. The *Session Key* value is used to distinguish between them.
- **Additional Encapsulation Checksum** - The GRE protocol allows for an additional checksum over and above the IPv4 checksum. This provides an extra check of data integrity.

The **Virtual Routing** options are used as with any other interface such as an Ethernet interface (see Section 4.3.2, “Ethernet Interfaces”). The routing tables specified here apply to the traffic carried by the tunnel and not the tunnel itself. The route lookup for the tunnel itself is specified in the earlier option **Outer PBR Table**.

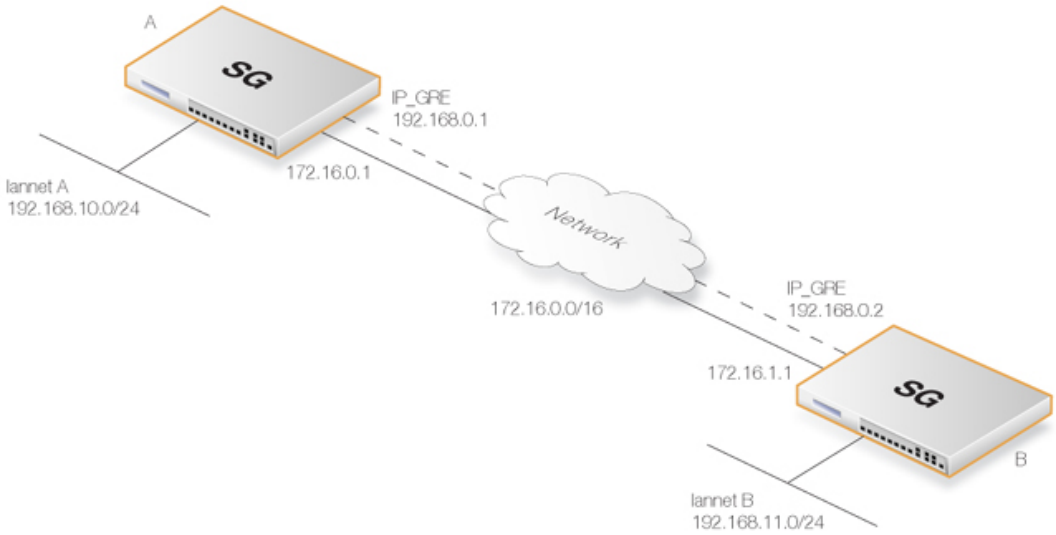
The **Advanced** settings for a GRE interface are:

- **Automatically add route for remote network** - This option would normally be checked in order that the routing table is automatically updated. The alternative is to manually create the required route.
- **Address to use as source IP** - It is possible to specify a particular IP address as the source interface IP for the tunnel.

GRE and the IP Rule Set

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 for 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.

An Example GRE Scenario



The diagram above shows a typical GRE scenario, where two Clavister Security Gateways **A** and **B** must communicate with each other through the intervening internal network *172.16.0.0/16*.

Any traffic passing between **A** and **B** is tunneled through the intervening network using a GRE tunnel and since the network is internal and not public there is no need for encryption.

Setup for Clavister Security Gateway *"A"*

Assuming that the network *192.168.10.0/24* is **lannet** on the **lan** interface, the steps for setting up CorePlus on **A** are:

1. In the address book set up the following IP objects:
   - **remote_net_B:** 192.168.11.0/24
   - **remote_gw:** 172.16.1.1
   - **ip_GRE:** 192.168.0.1
2. Create a GRE Tunnel object called **GRE_to_B** with the following parameters:
   - **IP Address:** ip_GRE
   - **Remote Network:** remote_net_B
   - **Remote Endpoint:** remote_gw
   - **Use Session Key:** 1
   - **Additional Encapulation Checksum:** Enabled
3. Define a route in the *main* routing table which routes all traffic to **remote_net_B** on the **GRE_to_B** GRE interface. This is not necessary if the option **Add route for remote network** is enabled in the **Advanced** tab, since this will add the route automatically.
4. Create the following rules in the IP rule set that allow traffic to pass through the tunnel:

Name	Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
To_B	Allow	lan	lannet	GRE_to_B	remote_net_B	All
From_B	Allow	GRE_to_B	remote_net_B	lan	lannet	All

Setup for Clavister Security Gateway *"B"*

Assuming that the network *192.168.11.0/24* is **lannet** on the **lan** interface, the steps for setting up CorePlus on **B** are as follows:

1. In the address book set up the following IP objects:
   - **remote_net_A:** 192.168.10.0/24
   - **remote_gw:** 172.16.0.1
   - **ip_GRE:** 192.168.0.2
2. Create a GRE Tunnel object called **GRE_to_A** with the following parameters:
   - **IP Address:** ip_GRE
   - **Remote Network:** remote_net_A
   - **Remote Endpoint:** remote_gw
   - **Use Session Key:** 1
   - **Additional Encapulation Checksum:** Enabled
3. Define a route in the *main* routing table which routes all traffic to **remote_net_A** on the **GRE_to_A** GRE interface. This is not necessary if the option **Add route for remote network** is enabled in the **Advanced** tab, since this will add the route automatically.
4. Create the following rules in the IP rule set that allow traffic to pass through the tunnel:

Name	Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
To_A	Allow	lan	lannet	GRE_to_A	remote_net_A	All
From_A	Allow	GRE_to_A	remote_net_A	lan	lannet	All

4.3.6. 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 security gateways. This is done by having more than one routing table so each routing table handles the routing for a virtual security gateway. In order to allow traffic to flow between one routing table and another, Loopback interfaces are needed.

Loopback Interfaces are configured with IP addresses, just as any other interface type but these IP addresses can be fictitious and do not need to be on the same network. The IP address given to a Loopback Interface 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* to *127.255.255.255*).

Example 4.13. Creating a Loopback Interface Pair

This example shows how to create a Loopback Interfaces pair.

Web Interface

A. Create the first loopback interface

1. Go to **Interfaces > Loopback Interfaces > Add**
2. Under **General** enter:
   - **Name:** enter a suitable name. For example *vr1-main*
   - **Loop to:** Leave this field blank for now
3. Under **IP Address** enter:
   - **IP Address:** the IP address of the interface
   - **Network:** the network for the interface
4. Click **OK**

B. Create the second loopback interface

1. Go to **Interfaces > Loopback Interfaces > Add**
2. Under **General** enter:
   - **Name:** enter a suitable name, for example *vr2-main*
   - **Loop to:** vr1-main
3. Under **IP Address** enter:
   - **IP Address:** the IP address of the interface
   - **Network:** the network for the interface
4. Click **OK**

C. Now go back to the first loopback interface and fill in the **Loop to:** field as *vr2-main*

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.

4.3.7. 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 example, of two Ethernet interfaces and four VLAN interfaces.

Example 4.14. Creating an Interface Group

CLI

```
Device:/> add Interface InterfaceGroup examplegroup Members=exampleif1,exampleif2
```

Web Interface

1. Go to **Interfaces > Interface Groups > Add > InterfaceGroup**
2. Enter the following information to define the group:
   - **Name:** The name of the group to be used later
   - **Security/Transport Equivalent:** 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

- **Interfaces:** Select the interfaces to be in the group

3. Click **OK**

4.4. ARP

4.4.1. 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. ARP operates at the OSI Data Link Layer (Layer 2 - see Appendix D, *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. Higher level protocols such as IP make use of IP addresses which are fundamentally different from a lower level hardware addressing scheme like the MAC address. ARP is used to retrieve the Ethernet MAC address of a host by using its IP address.

When a host needs to resolve an IP address to the corresponding Ethernet address, it broadcasts an ARP request packet. The ARP request packet contains the source MAC address, 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.

4.4.2. 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.

Clavister SG10 Ethernet Connection Limitations

With the Clavister SG10 Series hardware there is a limit to how many devices can be connected via the ethernet ports. This number is determined by the type of SG10 license purchased and the size of CorePlus ARP table that the license allows. There is no limitation on other Clavister hardware series.

4.4.3. 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 example, 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 advanced setting *ARP Expire*.

The setting *ARP Expire Unknown* specifies how long CorePlus will 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.

Example 4.15. Displaying the ARP Cache

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

CLI

```
Device:/> arp -show

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 delete all dynamic ARP entries from the cache, thereby forcing CorePlus to issue new ARP queries.

Example 4.16. Flushing the ARP Cache

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

CLI

```
Device:/> 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 security gateway, it might be necessary to adjust this value. This can be done by modifying the advanced setting *ARP Cache Size*.

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. The administrator can modify the advanced setting *ARP Hash Size* to reflect specific network requirements. The default value of this setting is *512*.

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

4.4.4. 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 4.17. 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:

CLI

```
Device:/> add ARP Interface=lan IP=192.168.10.15 Mode=Static
                        MACAddress=4b-86-f6-c5-a2-14
```

Web Interface

1. Go to **Interfaces > ARP > Add > ARP**
2. Select the following from the dropdown lists:
   - **Mode:** Static
   - **Interface:** lan
3. Enter the following:
   - **IP Address:** 192.168.10.15
   - **MAC:** 4b-86-f6-c5-a2-14
4. 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: Using Proxy ARP to publish networks

In the configuration of ARP entries, addresses may only be published one at a time. However, you can use the ProxyARP feature to handle publishing of entire networks (see Section 5.2.5, “Proxy ARP”).

4.4.5. ARP Advanced Settings Summary

The following advanced settings are available with ARP:

ARP Match Ethernet Sender

Determines if CorePlus will require the sender address at Ethernet level to comply with the hardware address reported in the ARP data.

Default: *DropLog*

ARP Query No Sender

Handles 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*

ARP Sender IP

Determines if the IP sender address must comply with the rules in the Access section.

Default: *Validate*

Unsolicited ARP Replies

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*

ARP Requests

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*

ARP Changes

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*

Static ARP Changes

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*

ARP Expire

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)*

ARP Expire Unknown

Specifies in seconds 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*

ARP Multicast

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*

ARP Broadcast

Determines how CorePlus deals with ARP requests and ARP replies that state that they are broadcast addresses. Such claims are usually never correct.

Default: *DropLog*

ARP cache size

How many ARP entries there can be in the cache in total.

Default: *4096*

ARP Hash Size

Hashing is used to rapidly look up entries in a table. For maximum efficiency, the hash size should be twice as large as the table it is indexing. If the largest directly-connected LAN contains 500 IP addresses then the size of the ARP entry hash should be at least 1000 entries.

Default: *512*

ARP Hash Size VLAN

Hashing is used to rapidly look up entries in a table. For maximum efficiency, the hash size should be twice as large as the table it is indexing, so if the largest directly-connected VLAN contains 500 IP addresses, the size of the ARP entry hash should be at least 1000 entries.

Default: *64*

ARP IP Collision

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*

4.5. The IP Rule Set

4.5.1. Security Policies

Common Policy Characteristics

CorePlus Security *Policies* designed by the administrator, regulate the way in which traffic can flow through the Clavister Security Gateway. Policies in CorePlus are defined by different CorePlus *rule sets*. These rule sets share a common means of specifying filtering criteria which determine the type of traffic to which they will apply. This set of criteria consists of:

Source Interface	An **Interface** or **Interface Group** where the packet is received at the Clavister Security Gateway. This can also be a VPN tunnel.
Source Network	The network that contains the source IP address of the packet. This might be a CorePlus IP object which could define a single IP address or range of addresses.
Destination Interface	An **Interface** or an **Interface Group** from which the packet would leave the Clavister Security Gateway. This can also be a VPN tunnel.
Destination Network	The network to which the destination IP address of the packet belongs. This might be a CorePlus IP object which could define a single IP address or range of addresses.
Service	The protocol type to which the packet belongs. Service objects define a protocol/port type. Examples might be **HTTP** or **ICMP**. Custom services can also be defined. See Section 4.2, “Services” for more information on this topic.

CorePlus Policy Rulesets

The principle CorePlus rule sets that define CorePlus security policies, and which use the same filtering parameters described above (networks/interfaces/service), include:

- **IP rules**
  These determine which traffic is permitted to pass through the Clavister Security Gateway as well as determining if the traffic is subject to address translation. They are described below.
- **Pipe rules**
  These determine which traffic triggers traffic shaping to take place and are described in Section 11.1, “Traffic Shaping”.
- **Policy-based Routing rules**
  These determine the routing table ro be used by traffic and are described in Section 5.3, “Policy-based Routing”.
- **IDP rules**
  These determine which traffic is subject to IDP scanning and are described in Section 7.5, “Intrusion Detection and Prevention”.
- **Authentication rules**
  These determine which traffic triggers authentication to take place (source net/interface only) and are described in Chapter 9, *User Authentication*.

Specifying Any Interface or Network

When specifying the filtering criteria in any of the rule sets specified above there are three useful pre-defined options that can be used :

- For a Source or Destination Network, the **all-nets** option is equivalent to the IP address *0.0.0.0/0* which will mean that any IP address is acceptable.
- For Source or Destination Interface, the **any** option can be used so that CorePlus will not care about the interface which the traffic is going to or coming from.
- The Destination Interface can be specified as **core**. This means that traffic, such as an ICMP *Ping*, is destined for the Clavister Security Gateway itself and CorePlus will respond to it.

IP Rules and the Default IP Rule Set

The *IP rule set* is the most important of these security policy rule sets. It determines the critical packet filtering function of CorePlus, regulating what is allowed or not allowed to pass through the Clavister Security Gateway, and if necessary, how address translations like NAT are applied.

There are two possible approaches to how traffic traversing the Clavister Security Gateway could be dealt with:

- Everything is denied unless specifically permitted
- **Or** everything is permitted unless specifically denied

To provide the best security, the first of these approaches is adopted by CorePlus. This means that when first installed and started, the CorePlus IP rule set drops all traffic. In order to allow any traffic to traverse the Clavister Security Gateway (including CorePlus responding to ICMP *Ping* requests), new IP rules must be defined by the administrator.

Traffic that does not match any rule in the IP rule set is, by default, dropped by CorePlus. For logging purposes it is nevertheless recommended that an explicit IP rule with an action of **Drop** for all source/destination networks/interfaces, and with logging enabled, is placed as the last rule in the IP rule set.

4.5.2. IP Rule Evaluation

When a new connection, such as a TCP/IP connection, is being established through the Clavister Security Gateway, the list of IP rules are evaluated from top to bottom until a rule that matches the parameters of the new connection is found. The rule's **Action** is then performed.

If the action allows it then the establishment of the new connection will go ahead. A new entry or *state* representing the new connection will then be added to the CorePlus internal *state table* which allows monitoring of opened and active connections passing through the Clavister Security Gateway. If the action is **Drop** or **Reject** then the new connection is refused.

Stateful Inspection

After initial rule evaluation of the opening connection, subsequent packets belonging to that connection will not need to be evaluated individually against the rule set. Instead, a highly efficient algorithm searches the state table for each packet to determine if it belongs to an established connection.

This approach is known as *stateful inspection* and is applied not only to stateful protocols such as TCP but also by means of "pseudo-connections" to stateless protocols such as UDP and ICMP. This approach means that evaluation against the IP rule set is only done in the initial opening phase of a connection. The size of the IP rule set consequently has negligible effect on overall throughput.

The First Matching Principle

If several rules match the same parameters, the first matching rule in a scan from top to bottom is the one that decides how the connection will be handled.

The exception to this is **SAT** rules since these rely on a pairing with a second rule to function. After encountering a matching **SAT** rule the search will therefore continue on looking for a matching second rule (see Section 8.3, “Static Address Translation” for more information on this).

Non-matching Traffic

Incoming packets that do not match any rule in the rule set and that do not have an already opened matching connection in the state table, will automatically be subject to a **Drop** action. For explicitness there should be a rule called **DropAll** as the final rule in the rule set with an action of **Drop** with Source/Destination Network *all-nets* and Source/Destination Interface *all*.

4.5.3. IP Rule Actions

A rule consists of two parts: the filtering parameters and the action to take if there is a match with those parameters. As described above, the parameters of any CorePlus rule, including IP rules are:

- Source Interface
- Source Network
- Destination Interface
- Destination Network
- Service

The *Service* in an IP rule is also important because if an *Application Layer Gateway* object is to be applied to traffic then it must be associated with a Service object (see Section 7.2, “ALGs”).

When an IP rule is triggered by a match then one of the following *Actions* can occur:

Allow 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 CorePlus "stateful engine".

FwdFast Let the packet pass through the Clavister Security Gateway without setting up a state for it in the state table. This means that the stateful inspection process is bypassed and is therefore less secure than **Allow** or **NAT** rules. Packet processing time is also slower than **Allow** rules since every packet is checked against the entire rule set.

NAT This functions like an **Allow** rule, but with dynamic address translation (NAT) enabled (see Section 8.1, “Dynamic Network Address Translation” in Chapter 8, *Address Translation* for a detailed description).

SAT This tells CorePlus to perform static address translation. A **SAT** rule always requires a matching **Allow**, **NAT** or **FwdFast** rule further down the rule set (see Section 8.3, “Static Address Translation” in Chapter 8, *Address Translation* for a detailed description).

Drop This tells CorePlus to immediately discard the packet. This is an "impolite" version of **Reject** in that no reply is sent back to the sender. It is often preferable since it gives a potential attacker no clues about what happened to their packets.

Reject This acts like **Drop**, but will return a "TCP RST" or "ICMP Unreachable message",

informing the sending computer that the packet was disallowed. This is a "polite" version of the **Drop** action.

Bi-directional Connections

A common mistake when setting up IP Rules is to define two rules, one rule for traffic in one direction and another rule for traffic coming back in the other direction. In fact nearly all IP Rules types allow *bi-directional* traffic flow once the initial connection is set up. The **Source Network** and **Source Interface** in the rule means the source of the initial connection request. If a connection is permitted and then becomes established, traffic can flow in either direction over it.

The exception to this bi-directional flow is **FwdFast** rules. If the **FwdFast** action is used, the rule will not allow traffic to flow from the destination back to the source. If bi-directional flow is required then two **FwdFast** rules are needed, one for either direction. This is also the case if a **FwdFast** rule is used with a **SAT** rule.

Using *Reject*

In certain situations the **Reject** action is recommended instead of the **Drop** action because a polite reply is required from CorePlus. An example of such a situation is when responding to the IDENT user identification protocol.

4.5.4. Multiple IP Rule Sets

Overview

CorePlus allows the administrator to define multiple IP rule sets which can both simplify and provide greater flexibility when defining security policies. The default IP rule set is known as *main* and is always present in CorePlus. Additional rule sets can be defined as needed and are given a name by the administrator.

Multiple IP rule sets offer many advantages, among them:

- The administrator can break a single large IP rule set into multiple, smaller and more manageable rule sets.
- A single named IP rule set can be associated with a routing table. This makes implementing Virtual Routing much simpler since each router can have a dedicated IP Rules-set associated with it. See Section 5.4, “Virtual Routing” for more information on this topic.

Searching Multiple rule sets

When multiple rule sets are defined, the way they are processed for a new connection is as follows:

- The primary *main* IP rule set is always searched first for matches of source/destination interface/network.
- User-defined rule sets are used in a rule look-up only when the action specified for a matching rule in *main* is **Goto**. The **Goto** action must have a named, administrator defined IP rule set associated with it and if the traffic matches the **Goto** rule then the rule look-up continues from the beginning of that named rule set.

  A **Goto** may never use the *main* rule set as its target.
- If the search in the named rule set finds no match then the connection is dropped.

- If a match is found in the named rule set then the action is executed. The action might be another **Goto** in which case the rule scanning jumps to the beginning of another named rule set.

  If the action is **Return** then the rule scanning resumes at the rule which follows the last **Goto** action (if there was no last **Goto** then the connection is dropped and rule scanning stops).

Loop Avoidance

It is conceivable that a sequence of **Goto** actions results in an infinite looping scanning sequence. CorePlus detects such logic when a configuration is initially uploaded. A new configuration is rejected if multiple IP rule set logic is detected that could potentially cause a loop condition.

The loop avoidance mechanism has to be efficient to enable fast configuration upload and for this reason it uses an algorithm that might sometimes find a fault in correct but complex logic. In this case it may be necessary to simplify the rule logic so the configuration can be uploaded.

A Usage Example

Below are two simple IP Rule set tables which illustrate how multiple rule sets might be used. The *main* rule set contains a first rule which has a **Goto** action which references the named administrator defined table called *ExtraRules*.

The administrator defined rule set *ExtraRules* contains a **NAT** and **SAT** rule. If neither are triggered then the final rule has a **Return** action which will cause the scanning process to go back to the rule in *main* which follows the **Goto** rule. In this case it will be the second rule in *main*.

The *main* IP rule set

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Service
1	Goto ExtraRules	any	all-nets	core	172.16.40.0/24	all
2	Allow	any	192.168.0.0/24	core	172.16.0.0/16	all

The *ExtraRules* IP rule set

#	Action	Src Iface	Src Net	Dest Iface	Dest Net	Service
1	SAT	any	all-nets	any	172.16.40.66	all
2	NAT	if2	176.16.0.0/16	any	all-nets	all
3	RETURN	if2	all-nets	any	all-nets	all

4.5.5. IP Rule Set Folders

In order to help organise large numbers of entries in IP rule sets, it is possible to create IP rule set *folders*. These folders are just like a folder in a computer's file system. They are created with a given name and can then be used to contain all the IP rules that are related together as a group.

Using folders is simply a way for the administrator to conveniently divide up IP rule set entries and no special properties are given to entries in different folders. CorePlus continues to see all entries as though they were in a single set of IP rules.

The folder concept is also used by CorePlus in the Address Book, where related IP address objects can be grouped together in administrator created folders.

Example 4.18. Adding an *Allow* IP Rule

This example shows how to create a simple *Allow* rule that will allow HTTP connections to opened from the *lannet* network on the *lan* interface to any network (*all-nets*) on the *wan* interface.

CLI

First, change the current category to be the *main* IP rule set:

```
Device:/> cc IPRuleSet main
```

Now, create the IP rule:

```
Device:/main> add IPRule Action=Allow Service=http SourceInterface=lan
                         SourceNetwork=lannet DestinationInterface=wan
                         DestinationNetwork=all-nets Name=lan_http
```

Return to the top level:

```
Device:/main> cc
```

Configuration changes must be saved by then issuing an *activate* followed by a *commit* command.

Web Interface

1. Go to **Rules > IP Rules > Add > IPRule**
2. Specify a suitable name for the rule, for example *LAN_HTTP*
3. Now enter:
   - **Name:** A suitable name for the rule. For example *lan_http*
   - **Action:** Allow
   - **Service:** http
   - **Source Interface:** lan
   - **Source Network:** lannet
   - **Destination Interface:** wan
   - **Destination Network:** all-nets
4. Click **OK**

4.6. 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.

Schedule Objects

CorePlus addresses this requirement by providing *Schedule* objects, or simply *schedules*, that can be selected and used with various types of security 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.

Multiple Time Ranges

A Schedule object also offers 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: Set the system date and time

As schedules depend on an accurate system date and time, it is very important that the system date and time are set correctly. This is also important for some other features such as certificate usage in VPN tunnels.

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 4.8, "Date and Time".

Example 4.19. 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.

CLI

```
Device:/> add ScheduleProfile OfficeHours Mon=8-17 Tue=8-17 Wed=8-17 Thu=8-17
     Fri=8-17
```

Now create the IP rule that uses this schedule. First, change the current category to be the *main* IP rule set:

```
Device:/> cc IPRuleSet main
```

Now, create the IP rule:

```
Device:/main> add IPRule Action=NAT Service=http SourceInterface=lan
     SourceNetwork=lannet DestinationInterface=any
     DestinationNetwork=all-nets Schedule=OfficeHours
     name=AllowHTTP
```

Return to the top level:

```
Device:/main> cc
```

Configuration changes must be saved by then issuing an *activate* followed by a *commit* command.

Web Interface

1. Go to **Objects > Schedules > Add > Schedule**
2. Enter the following:
   - **Name:** OfficeHours
3. Select 08-17, Monday to Friday in the grid
4. Click **OK**

1. Go to **Rules > IP Rules > Add > IPRule**
2. Enter the following:
   - **Name:** AllowHTTP
3. Select the following from the dropdown lists:
   - **Action:** NAT
   - **Service:** http
   - **Schedule:** OfficeHours
   - **SourceInterface:** lan
   - **SourceNetwork** lannet
   - **DestinationInterface:** any
   - **DestinationNetwork:** all-nets
4. Click **OK**

4.7. Certificates

4.7.1. Overview

X.509

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. References in this manual to a *certificate* means a *X.509 certificate*.

A 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 fake key with the name and user ID of an intended recipient.

Certificates with VPN Tunnels

The main usage of certificates in CorePlus is with VPN tunnels. The simplest and fastest way to provide security between the ends of a tunnel is to use Pre-shared Keys (PSKs). As a VPN network grows so does the complexity of using PSKs. Certificates provide a means to better manage security in much larger networks.

Certificate Components

A certificate consists of the following:

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

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.

Certifice Authorities

A *certificate 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 CA 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.

Note
A CA is sometimes referred to as a "certification authority".

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.

Important
Make sure the CorePlus date and time are set correctly when using certificates.

Certificate Revocation Lists

A *Certificate Revocation List* (CRL) contains a list of all certificates that have been cancelled before their expiration date. They are normally held on an external server which is accessed to determine if the certificate is still valid. The ability to validate a user certificate in this way is a key reason why certificate security simplifies the administration of large user communities.

CRLs are published on servers that all certificate users can access, using either the LDAP or HTTP protocols. Revocation 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, perhaps because they have left the company. Whatever the reason, server CRLs can be updated to change the validity of one or many certificates.

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.

A CA usually 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.

Trusting Certificates

When using certificates, CorePlus 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.

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.

Reusing Root Certificates

In CorePlus, root certificates should be seen as global entities that can be reused between VPN tunnels. Even though a root certificate is associated with one VPN tunnel in CorePlus, it can still be reused with any number of other, different VPN tunnels.

4.7.2. Certificates in CorePlus

Certificates can be uploaded to CorePlus 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. Self-signed certificates can be generated by using one of a number of freely available utilities for doing this.

Example 4.20. Uploading a Certificate

The certificate may either be self-signed or belonging to a remote peer or CA server.

Web Interface

1. Go to **Objects > Authentication Objects > Add > Certificate**
2. Specify a suitable name for the certificate
3. Now select one of the following:
   - **Upload self-signed X.509 Certificate**
   - **Upload a remote certificate**
4. Click **OK** and follow the instructions

Example 4.21. Associating Certificates with IPsec Tunnels

To associate an imported certificate with an IPsec tunnel.

Web Interface

1. Go to **Interfaces > IPsec**
2. Display the properties of the IPsec tunnel
3. Select the **Authentication** tab
4. Select the **X509 Certificate** option
5. Select the correct **Gateway** and **Root** certificates
6. Click **OK**

4.7.3. CA Certificate Requests

To request certificates from a CA server or CA company, the best method is to send a *CA Certificate Request* which is a file that contains a request for a certificate in a well known, pre-defined format.

Manually Creating Windows CA Server Requests

The CorePlus Web Interface (WebUI) does not currently include the ability to generate certificate requests that can be sent to a CA server for generation of the *.cer* and *.key* files required by CorePlus.

It is possible, however, to manually create the required files for a Windows CA server using the following stages.

- Create a *gateway certificate* on the Windows CA server and export it as a file in the *.pfx* format.

- Convert the *.pfx* file into the *.pem* format.
- Take out the relevant parts of the *.pem* file to form the required *.cer* and *.key* files.

The detailed steps for the above stages are as follows:

1. Create the *gateway certificate* on the Windows CA server and export it to a *.pfx* file on the local CorePlus management workstation disk.
2. Now convert the local *.pfx* file to a *.pem* file. This can be done with the *OpenSSL* utility using the console command line:

   ```
   openssl pkcs12 -in gateway.pfx -out gateway.pem -nodes
   ```

   In this command line example, the file exported from the CA server is assumed to be called *gateway.pfx* and it is assumed to be in the same local directory as the OpenSSL executable.

   The original *gateway.pfx* file contained 3 certificates: CA root certificate, a personal certificate and a private key certificate. The *gateway.pem* file now contains these in format which can be cut and pasted with a text editor.

   Note
   OpenSSL is being used here as a conversion utility and not in its normal role as a communication utility.

3. Create two blank text files with a text editor, such as Windows Notepad. Give the files the same filename but use the extension *.cer* for one and *.key* for the other. For example, *gateway.cer* and *gateway.key* might be the names.
4. Start a text editor and open the downloaded *.pem* file and locate the line that begins:

   ```
   -----BEGIN RSA PRIVATE KEY-----
   ```

5. Mark and copy into the system clipboard that line and everything under it, up to and including the line:

   ```
   -----END RSA PRIVATE KEY-----
   ```

6. Now paste the copied text into the *.key* file and save it.
7. Back in the *.pem* file, locate the line that begins:

   ```
   -----BEGIN CERTIFICATE-----
   ```

   and copy into the system clipboard that line and everything under it, up to and including:

   ```
   -----END CERTIFICATE-----
   ```

8. Now paste this copied text into the *.cer* file and save it.

The saved *.key* and *.cer* files are now ready for upload into CorePlus.

4.8. Date and Time

4.8.1. Overview

Correctly setting the date and time is important for CorePlus to operate properly. Time scheduled policies, auto-update of the IDP and Anti-Virus databases, and other product features require that the system clock is accurately set. 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 correctly synchronized with other devices in the network.

To maintain current date and time, CorePlus makes use of a built-in real-time hardware clock. This clock is also equipped with a battery backup to guard against a temporary loss of power. In addition, CorePlus supports *Time Synchronization Protocols* in order to automatically adjust the clock, based on queries sent to special external servers.

4.8.2. Setting Date and Time

Current Date and Time

The administrator can set the date and time manually and this is recommended when a new CorePlus installation is started for the first time.

Example 4.22. Setting the Current Date and Time

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

CLI

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

Where YYYY-mm-DD HH:MM:SS is the new date and time. Note that the date order is year, then month and then day. For example, to set the date and time to 9:25 in the morning on April 27th, 2008 the command would be:

```
Device:/> time -set 2008-04-27 09:25:00
```

Web Interface

1. Go to **System > Date and Time**
2. Click **Set Date and Time**
3. Set year, month, day and time via the dropdown controls
4. Click **OK**

Note

A new date and time will be applied by CorePlus as soon as it is set. There is no need to reconfigure or restart the system

Time Zones

The world is divided up into a number of time zones with Greenwich Mean Time (GMT) in London at zero longitude being taken as the base time zone. All other time zones going east and west from zero longitude are taken as being GMT plus or minus a given integer number of hours. All locations counted as being inside a given time zone will then have the same local time and this will be one of the integer offsets from GMT.

The CorePlus time zone setting reflects the time zone where the Clavister Security Gateway is physically located.

Example 4.23. Setting the Time Zone

To modify the CorePlus time zone to be GMT plus 1 hour, follow the steps outlined below:

CLI

```
Device:/> set DateTime Timezone=GMTplus1
```

Web Interface

1. Go to **System > Date and Time**
2. Select **(GMT+01:00)** in the **Timezone** drop-down list
3. Click **OK**

Daylight Saving Time

Many regions follow *Daylight Saving Time* (DST) (or "Summer-time" as it is called in some countries) and this means clocks are advanced for the summer period. Unfortunately, the principles regulating DST vary from country to country, and in some cases there can be variations within the same country. For this reason, CorePlus 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 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. The DST offset indicates the number of minutes to advance the clock during the daylight saving time period.

Example 4.24. Enabling DST

To enable DST, follow the steps outlined below:

CLI

```
Device:/> set DateTime DSTEnabled=Yes
```

Web Interface

1. Go to **System/Date and Time**
2. Check **Enable daylight saving time**
3. Click **OK**

4.8.3. Time Servers

The hardware clock which CorePlus uses can sometimes become fast or slow after a period of operation. This is normal behavior in most network and computer equipment and is solved by utilizing *Time Servers*.

CorePlus is able to adjust the clock automatically based on information received from one or more Time Servers which provide a highly accurate time, usually using atomic clocks. Using Time Servers is highly recommended as it ensures CorePlus will have its date and time aligned with other network devices.

Time Synchronization Protocols

Time Synchronization Protocols are standardized methods for retrieving time information from external Time Servers. CorePlus supports the following time synchronization protocols:

- SNTP - Defined by RFC 2030, The Simple Network Time Protocol (SNTP) is a lightweight implementation of NTP (RFC 1305). This is used by CorePlus to query NTP servers.
- 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 server sends back the time in seconds since midnight on January first, 1900.

Most public Time Servers run the NTP protocol and are accessible using SNTP.

Configuring Time Servers

Up to three Time Servers can be configured to query for time information. By using more than a single server, situations where an unreachable server causes the time synchronization process to fail can be prevented. CorePlus always queries all configured Time Servers and then computes an average time based on all responses. Internet search engines can be used to list publicly available Time Servers.

Important

Make sure an external DNS server is configured so that Time Server URLs can be resolved (see Section 4.9, "DNS"). This is not needed if using server IP addresses.

Example 4.25. Enabling Time Synchronization using SNTP

In this example, time synchronization is set up to use the SNTP protocol to communicate with the NTP servers at the Swedish National Laboratory for Time and Frequency. The NTP server URLs are *ntp1.sp.se* and *ntp2.sp.se*.

CLI

```
Device:/> set DateTime TimeSynchronization=custom TimeSyncServer1=dns:ntp1.sp.se
     TimeSyncServer2=dns:ntp2.sp.se TimeSyncInterval=86400
```

Web Interface

1. Go to **System > Date and Time**
2. Check the **Enable time synchronization**
3. Now enter:
   - **Time Server Type:** SNTP
   - **Primary Time Server:** ntp1.sp.se
   - **Secondary Time Server:** ntp2.sp.se
4. Click **OK**

Example 4.26. Manually Triggering a Time Synchronization

Time synchronization can be triggered from the CLI. The output below shows a typical response.

CLI

```
Device:/> time -sync
Attempting to synchronize system time...

Server time: 2008-02-27 12:21:52 (UTC+00:00)
Local time:  2008-02-27 12:24:30 (UTC+00:00) (diff: 158)

Local time successfully changed to server time.
```

Maximum Time Adjustment

To avoid situations where a faulty Time Server causes the clock to be updated with a extremely inaccurate time, a *Maximum Adjustment* value (in seconds) can be set. If the difference between the current CorePlus time and the time received from a Time Server is greater than this Maximum Adjustment value, then the Time Server response will be discarded. For example, assume that the maximum adjustment value is set to 60 seconds and the current CorePlus time is 16:42:35. If a Time Server responds with a time of 16:43:38 then the difference is 63 seconds. This is greater than the Maximum Adjustment value so no update occurs for this response.

Example 4.27. Modifying the Maximum Adjustment Value

CLI

```
Device:/> set DateTime TimeSyncMaxAdjust=40000
```

Web Interface

1. Go to **System > Date and Time**
2. For the setting **Maximum time drift that a server is allowed to adjust**, enter the maximum time drift in seconds that a server is allowed to adjust for
3. Click **OK**

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

Example 4.28. Forcing Time Synchronization

This example demonstrates how to force time synchronization, overiding the maximum adjustment setting.

CLI

```
Device:/> time -sync -force
```

Synchronization Intervals

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 in a 24 hour period.

4.8.4. Settings Summary for Date and Time

Below is a summary of the various settings for date and time:

Time Zone

Time zone offset in minutes.

Default: *0*

DST Offset

Daylight saving time offset in minutes.

Default: *0*

DST Start Date

What month and day DST starts, in the format MM-DD.

Default: *none*

DST End Date

What month and day DST ends, in the format MM-DD.

Default: *none*

Time Sync Server Type

Type of server for time synchronization, UDPTime or SNTP (Simple Network Time Protocol).

Default: *SNTP*

Primary Time Server

DNS hostname or IP Address of Timeserver 1.

Default: *None*

Secondary Time Server

DNS hostname or IP Address of Timeserver 2.

Default: *None*

teriary Time Server

DNS hostname or IP Address of Timeserver 3.

Default: *None*

Interval between synchronization

Seconds between each resynchronization.

Default: *86400*

Max time drift

Maximum time drift in seconds that a server is allowed to adjust.

Default: *600*

Group interval

Interval according to which server responses will be grouped.

Default: *10*

4.9. DNS

Overview

A DNS server can resolve a *Fully Qualified Domain Name* (FQDN) into the corresponding numeric IP address. FQDNs are unambiguous textual domain names which specify a node's unique position in the Internet's DNS tree hierarchy. FQDN resolution allows the actual physical IP address to change while the FQDN can stay the same.

A *Uniform Resource Locator* (URL) differs from an FQDN in that the URL includes the access protocol along with the FQDN. For example the protocol might be specified *http//:* for world wide web pages.

FQDNs are used in many aspects of a CorePlus configuration where IP addresses are unknown or where it makes more sense to make use of DNS resolution instead of using static IP addresses.

DNS with CorePlus

To accomplish DNS resolution, CorePlus has a built-in DNS client that can be configured to make use of up to three DNS servers. The are called the *Primary Server*, the *Secondary Server* and the *Tertiary Server*. For DNS to function at least the primary must be defined. It is recommended to have at least a primary and secondary defined so that there is a backup should the primary be unavailable.

Features Requiring DNS Resolution

Having at least one DNS server defined is vital for functioning of the following modules in CorePlus:

- Automatic time synchronization.
- Access to an external certificate authority server for CA signed certificates.
- UTM features that require access to external servers such as anti-virus and IDP.

Example 4.29. 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.

CLI

```
Device:/> set DNS DNSServer1=10.0.0.1 DNSServer2=10.0.0.2
```

Web Interface

1. Go to **System > DNS**
2. Enter the following:
   - **Primary Server:** 10.0.0.1
   - **Secondary Server:** 10.0.0.2
3. Click **OK**

Dynamic DNS

A DNS feature offered by CorePlus is the ability to explicitly inform DNS servers when the external IP address of the Clavister Security Gateway has changed. This is sometimes referred to as *Dynamic DNS* and is useful where the Clavister Security Gateway has an external IP address that can change.

Dynamic DNS can also be useful in VPN scenarios where both ends of the tunnel have dynamic IP addresses. If only one side of the tunnel has a dynamic address then the CorePlus VPN *keep alive* feature solves this problem.

Under **System > Misc. Clients** in the WebUI, several dynamic DNS services are defined. The *HTTP Poster* client is a generic dynamic DNS client with which it is possible to define 3 different DNS URLs plus an explicit value for *Delay in seconds until all URLs are refetched* (with a default of 604800 seconds, equivalent to 7 days).

Everytime CorePlus reconfigures, *HTTP Poster* will send an HTTP *GET* request to the defined URLs. It will also repeat this at an interval specified by the refetch delay.

The difference between *HTTP Poster* and the named DNS servers in the WebUI is that *HTTP Poster* can be used to send any URL. The named services are a convenience that make it easy to correctly format the URL needed for that service. For example, the http:// URL for the *dyndns.org* service might be:

```
myuid:mypwd@members.dyndns.org/nic/update?hostname=mydns.dyndns.org
```

This could be sent as shown above by using *HTTP Poster*, or the URL could be automatically formatted for the administrator by CorePlus through choosing the *DynDNS* menu option and entering the information required for *dyndns.org*.

The CLI console command *httpposter* can be used to troubleshoot problems by seeing what CorePlus is sending and what the servers are returning.

Note

Dynamic DNS services are often sensitive to repeated logon attempt over short periods of time and may blacklist IP addresses that are sending excessive requests. It is therefore not advisable to query these services too often otherwise they may cease to respond.

HTTP Poster may be used for other purposes than dynamic DNS. Any need for CorePlus to generate an HTTP *GET* request can be met by the feature.

4.10. Internet Access Setup

Overview

One of the first things an administrator often wants to do after starting CorePlus for the first time is to set up access to the public Internet. This section is a quick start guide to doing this and refers to many other sections in Chapter 4, *Fundamentals* as well as Chapter 5, *Routing* which follows.

IP Address Options

Initially, access to the Internet will not be possible via a Clavister Security Gateway. To achieve this, access must first be arranged with an Internet Service Provider (ISP). The ISP may offer two options for access:

- **Using Static Addresses**
  Access can be setup with static IP addresses where the ISP provides the required IP addresses (for example, in an email) and these are manually input into CorePlus.
- **Using DHCP**
  CorePlus can act as a DHCP client, using the DHCP protocol to automatically retrieve all the IP addresses required across the connection with the ISP.

4.10.1. Static Address Setup

For the static IP address option, the ISP should provide:

- A public IP address for the Clavister Security Gateway.
- The IP address of the ISPs "gateway" router.
- Optionally, a network address for the network between the ISP and the Clavister Security Gateway. This can be used for addressing any hosts that lie on this network aside from the ISPs gateway.

CorePlus Ignores the Broadcast Address
The broadcast IP address need not be specified. It is automatically calculated by CorePlus but is not responded to.

Creating IP Objects

Once the ISP has provided the necessary information, several *IP objects* in the CorePlus *Address Book* need to be defined. The Address Book provides a way to associate a text name with an IP address, IP range or IP network. Instead of re-typing IP addresses, these names can then be used as a choice throughout the Web Interface as well as being used with CLI commands.

Before defining any new addresses, the Address Book already contains a number of useful addresses by default. One of the most useful is the *all-nets* address, which corresponds to the IP address *0.0.0.0/0* (all hosts and networks).

To create the IP objects necessary for internet access, go to **Objects** > **Address Book** in the Web Interface navigation tree and then select **Add** > **IP address**.

The following IP objects should be added or modified:

- Create a new IP object called *gw-world* with the IP address of the ISP gateway.

- Modify the predefined object *ip_wan* so it is assigned the public IP address of the Clavister Security Gateway.
- If the network between the Clavister Security Gateway and ISP is to be addressed then the object *wannet* should be allocated the network address provided by the ISP.

Interface Naming

The physical interface *wan* is assumed in this manual to be the default name of the interface to which the ISP is connected. Different hardware models can have different default names and these names my be changed by the administrator. The above should therefore be adapted according to the particular hardware in use.

CorePlus uses certain naming conventions for particular objects, such as *gw-world* for the ISP gateway. These need not be followed by the administrator but if they are followed, it makes examination of a configuration by Clavister support personnel easier.

The contents of the Address Book can be examined using the CLI command:

```
Device:/> show Address IP4Address
```

4.10.2. DHCP Setup

The IP addresses defined in the previous section can be retrieved automatically from the ISP using the DHCP protocol. The steps to enable this are:

- In the Web Interface navigation tree, go to **Interfaces** > **Ethernet** and select the name of the interface that will be used for ISP access. The properties for that interface will be displayed.
- Check the option to *Enable DHCP on this interface*. This will automatically populate the Address Book with all relevant IP addresses when they are received from the ISP via DHCP.

See Chapter 6, *DHCP Services* for more information on this topic.

4.10.3. Creating Routes

At this stage no route exists in the *main* routing table that allows traffic to reach the Internet so this must be defined. The routes parameters will be as follows:

Interface	Network	Gateway
wan	all-nets	gw-world

To do this in the Web Interface go to **Routes** > **New Route** in the navigation tree.

The route to access *wannet* via the *wan* interface will be automatically created when the *wannet* object is edited.

See Chapter 5, *Routing* for more information on this topic.

4.10.4. Creating IP Rules

Before traffic can flow to the ISP, appropriate *IP Rules* must be created in the IP rule set to allow the traffic to pass.

At minimum, DNS and HTTP traffic should be allowed to flow so that web surfing can take place. It is also recommended to use NAT to share the single external IP address assigned to the Clavister Security Gateway so that the internal network topology is hidden. If, for example, web surfing is

going to be done from several hosts on the internal network attached to the *lan* interface then the IP rules for DNS and HTTP would be:

Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
NAT	lan	lannet	any	all-nets	dns-all
NAT	lan	lannet	any	all-nets	http-all

The service *http-all* includes both the HTTP and HTTPS services. The single service *all* could have been used in a single rule but this is not recommended as this means connections could be opened on any port number which can compromise security. The best approach is to define the filter for traffic as narrowly as possible which is what has been done here.

See Section 4.5, “The IP Rule Set” for more information on this topic.

4.10.5. Defining DNS Servers

Web surfing cannot function properly without the ability to resolve HTTP URLs through looking up these URLs on DNS servers.

In the Web Interface go to **System > DNS** of the navigation tree. Up to three DNS servers can be manually specified here and at least one should be configured for the DNS lookup to function.

Automatic Configuration through DHCP

If connection to an ISP is done with DHCP then the ISP's DHCP server should automatically configured at least one DNS server in CorePlus at the same time all other IP addresses are cinfgured during initial connection to the ISP.

When DHCP configures the DHCP servers in CorePlus, names are automatically assigned to these servers so they can be referenced later in user interfaces. The names used are of the form *<if-name>_dns1*, *<if-name>_dns2* and so on, where *<if-name>* is the interface name on which the DHCP addresses are received.

When CorePlus itself acts as a DHCP server, the DNS addresses it hands out are, by default, the same as those received through DHCP unless different DNS servers are explicitly specified.

Default DNS Server Address Names with DHCP

When CorePlus receives DNS servers through DHCP, these are automatically added to the Address Book with a default name that follows the pattern *<interface-name>_DNS<num>*. For example, the second DNS server address allocated though the *wan* interface would appear in the Address Book as *wan_DNS2*.

When viewed in the these DNS addresses always have the value *0.0.0.0*. To see the actual value after it has been assigned with DHCP, use the CLI command *dhcp -show*. For example, to see the status of the DNS servers on the *wan* interface, use the command:

```
Device:/> dhcp -show wan
```

See Section 4.9, “DNS” for more information on this topic.

Chapter 5. Routing

This chapter describes how to configure IP routing in CorePlus.

5.1. Overview

IP routing is one of the most fundamental functions of CorePlus. Any IP packet flowing through a Clavister Security Gateway will be subjected to at least one routing decision at some point in time, and properly setting up routing is crucial for the system to function as expected.

CorePlus offers support for the following types of routing mechanisms:

- Static routing
- Dynamic routing
- Virtual routing

CorePlus additionally supports *route monitoring* to achieve route and link redundancy with fail-over capability.

5.2. 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.

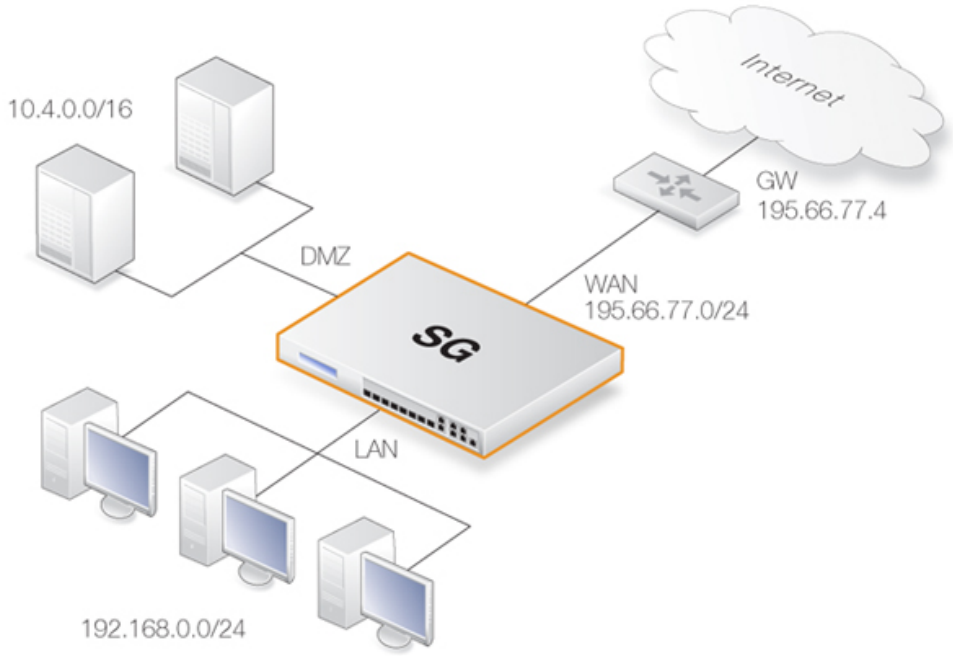
For more information about the dynamic routing capabilities of CorePlus, please see Section 5.5, “Dynamic Routing”. Note however, that even if you choose to implement dynamic routing for your network, you will still need to understand the principles of static routing and how it is implemented in CorePlus.

5.2.1. The Principles of Routing

IP routing is the mechanism used in TCP/IP based networks 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 consists of:

- A destination network.
- The interface to forward the packet on to reach that network.
- Optionally, the IP address of the *gateway* which is the next router in the path to the destination.

The diagram below illustrates a typical Clavister Security Gateway deployment:



In the above diagram, the **LAN** interface is connected to the network *192.168.0.0/24* and the **DMZ** interface is connected to the network *10.4.0.0/16*. The **WAN** interface is connected to the network *195.66.77.0/24* and the address of the ISP gateway to the public Internet is *195.66.77.4*.

The associated routing table for this would be as follows:

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	all-nets	195.66.77.4

The above 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 *all-nets* 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 *all-nets* 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.

5.2.2. Static Routing

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

CorePlus supports multiple routing tables. A default table called **main** is pre-defined and is always present in CorePlus. However, additional and completely separate routing tables can be defined by the administrator to provide alternate routing.

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

- **Virtual Routing** associates interfaces with a particular routing table. This enables a single physical Clavister Security Gateway to act as multiple virtual systems. Communication between these systems is achieved with *Loopback Interfaces* (see Section 5.4, “Virtual Routing” and also Section 4.3.6, “Loopback Interfaces”).
- **Policy Based Routing Rules** can be defined which decide which of the routing tables will deal with certain types of traffic (see Section 5.3, “Policy-based Routing”).

The Route Lookup Mechanism

The CorePlus route lookup mechanism has some slight differences to how some other router products work. In many routers, where the IP packets are forwarded without context (in other words, the forwarding is stateless), the routing table is scanned for each and every IP packet received by the router. In CorePlus, packets are forwarded with state-awareness, so the route lookup process is tightly integrated into the CorePlus stateful inspection mechanism.

When an IP packet is received on any of the interfaces, the connection table is consulted to see if

there is an already open connection for which the received packet belongs. If an existing connection is found, the connection table entry includes information on where to route the packet so there is no need for lookups in the routing table. This is far more efficient than traditional routing table lookups, and is one reason for the high forwarding performance of CorePlus.

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

CorePlus Route Notation

CorePlus uses a slightly different way of describing routes compared to most other systems but this way is easier to understand, making errors less likely.

Many other products do not use the specific interface in the routing table, but specify the IP address of the interface instead. The routing table below 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 is 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
```

The CorePlus way of describing the routes is easier to read and understand. 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 to 192.168.0.17 and another route for addresses 192.168.0.18 to 192.168.0.254. This is a feature that makes CorePlus highly suitable for routing in highly 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 5.1. Displaying the Routing Table

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

CLI

To see the configured routing table:

```
Device:/> cc RoutingTable main

Device:/main> show

Route
   #  Interface  Network   Gateway        Local IP
   -  ---------  --------  -------------  --------
   1  wan        all-nets  213.124.165.1  (none)
   2  lan        lannet    (none)         (none)
   3  wan        wannet    (none)         (none)
```

To see the active routing table enter:

```
Device:/> 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
```

Web Interface

To see the configured routing table:

1. Go to **Routing > Routing Tables**
2. Select and right-click the *main* routing table in the grid

The main window will list the configured routes

To see the active routing table in the Web Interface, select the **Routes** item in the **Status** dropdown menu in the menu bar - the main window will list the active routing table

Tip

Notice that in the CLI example above, it was necessary to first select the name of a specific RoutingTable with the "cc" command before manipulating individual routes. This is necessary for any category that could contain more than one named group of objects.

Initial Static Routes

When the Clavister Security Gateway is configured for the first time, the routing table will have one route defined for each interface. The routes will have default IP addresses which must be changed to the appropriate IP address ranges for traffic to flow.

The most important route that must be defined is the route to *all-nets* which should correspond with the ISP and public Internet access. If using the CorePlus setup wizard, this route is added automatically.

The option exists on any interface to indicate that it is the interface for connection to the internet. When this option is selected the *all-nets* route is automatically added to the *main* routing table for the interface.

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	

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. In other words, it is processed by CorePlus 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 5.2. Displaying the Core Routes

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

CLI

```
Device:/> 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
```

Web Interface

1. Select the **Routes** item in the **Status** dropdown menu in the menu bar
2. Check the **Show all routes** checkbox and click the **Apply** button
3. The main window will list the active routing table, including the core routes

Tip
*For detailed information about the output of the CLI **routes** command. Please see the CLI Reference Guide.*

5.2.3. Route Failover

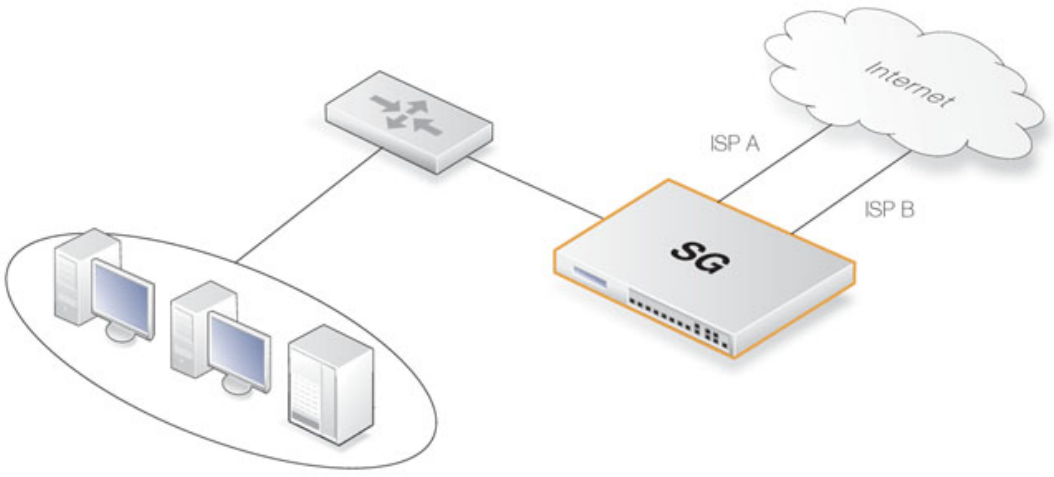
Overview

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 disrupted if an Internet connection fails.

As a consequence, it is quite common to have backup Internet connectivity using a secondary Internet Service Provider (ISP). The connections to the two service providers often use different access methods to avoid a single point of failure.

To allow for a situation with, for example, multiple ISPs, CorePlus provides a *Route Failover* capability so that should one route fail, traffic can automatically failover to another, alternate route. CorePlus implements Route Failover through the use of *Route Monitoring* in which CorePlus monitors the availability of routes and then switches traffic to an alternate route should the primary, preferred one fail.

Figure 5.1. A Route Failover Scenario for ISP Access



Setting Up Route Failover

Route Monitoring should be enabled on a per-route basis. To enable the Route Failover feature in a scenario with a preferred and a backup route, the preferred route will have Route Monitoring enabled, however the backup route does not require it to be enabled since it will usually have no route to failover to. For a route with Route Monitoring enabled, one of two Route Monitoring

methods must be chosen:

Interface Link Status CorePlus 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 for monitoring 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 specific gateway has been specified as the next hop for a route, accessibility to that gateway can be monitored by sending periodic ARP requests. As long as the gateway responds to these requests, the route is considered to be functioning correctly.

Host Monitoring The first two options check the accessibility of components local to the Clavister Security Gateway. An alternative is to monitor the accessibility of one or more nominated remote hosts. These hosts might have known high availability and polling them can indicate if traffic from the local Clavister Security Gateway is reaching them. Host monitoring also provides a way to measure the network delays in reaching remote hosts and to initiate failover to an alternate route if delays exceed administrator-specified thresholds.

Setting the Route Metric

When specifying routes, the administrator should manually set a route's *Metric*. The Metric is a positive integer that indicates how preferred the route is as a means to reach its destination. When two routes offer a means to reach the same destination, CorePlus will select the one with the lowest Metric value for sending data (if two routes have the same Metric, the route found first in the routing table will be chosen).

A primary, preferred route should have a lower Metric (for example "10"), and a secondary, failover route should have a higher Metric value (for example "20").

Multiple Failover Routes

It is possible to specify more than one failover route. For instance, the primary route could have two other routes as failover routes instead of just one. In this case the Metric should be different for each of the three routes: "10" for the primary route, "20" for the first failover route and "30" for the second failover route. The first two routes would have Route Monitoring enabled in the routing table but the last one (with the highest Metric) would not since it has no route to failover to.

Failover Processing

Whenever monitoring determines that a route is not available, CorePlus will mark the route as disabled and instigate Route Failover for existing and new connections. For already established connections, a route lookup will be performed to find the next best matching route and the connections will then switch to using the new route. For new connections, route lookup will ignore disabled routes and the next best matching route will be used instead.

The table below defines two default routes, both having *all-nets* as the destination, but using two different gateways. The first, primary route has the lowest Metric and also has Route Monitoring enabled. Route Monitoring for the second, alternate route is not meaningful since it has no failover route.

Route #	Interface	Destination	Gateway	Metric	Monitoring
1	wan	all-nets	195.66.77.1	10	On
2	wan	all-nets	193.54.68.1	20	Off

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

Re-enabling Routes

Even if a route has been disabled, CorePlus will continue to check the status of that route. Should the route become available again, it will be re-enabled and existing connections will automatically be transferred back to it.

Route Interface Grouping

When using route monitoring, it is important to check if a failover to another route will cause the routing interface to be changed. If this could happen, it is necessary to take some precautionary steps to ensure that policies and existing connections will be maintained.

To illustrate the problem, consider the following configuration:

First, there is 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

The routing table consequently contains the following default route:

Route #	Interface	Destination	Gateway	Metric	Monitoring
1	wan	all-nets	195.66.77.1	10	Off

Now a secondary route is added over a backup DSL connection and Route Monitoring is enabled for this. The updated routing table will look like this:

Route #	Interface	Destination	Gateway	Metric	Monitoring
1	wan	all-nets	195.66.77.1	10	On
2	dsl	all-nets	193.54.68.1	20	Off

Notice that Route Monitoring is enabled for the first route but not the backup, failover route.

As long as the preferred **wan** route is healthy, everything will work as expected. Route Monitoring will also be functioning, so the secondary route will be enabled if the **wan** route should fail.

There are, however, some problems with this setup: if a route failover occurs, the default route will then use the **dsl** interface. When a new HTTP connection is then 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 the original NAT rule assumes the destination interface to be **wan** so the new connection will be dropped by the rule set.

In addition, any existing connections matching the NAT rule will also be dropped as a result of the change in the destination interface. Clearly, this is undesirable.

To overcome this issue, potential destination interfaces should be grouped together into an *Interface Group* and the **Security/Transport Equivalent** flag should be enabled for the Group. The Interface Group is then used as the Destination Interface when setting policies. For more information on groups, see Section 4.3.7, “Interface Groups”.

Gratuitous ARP Generation

By default CorePlus generates a gratuitous ARP request when a route failover occurs. The reason

for this is to notify surrounding systems that there has been a route change. This behavior can be controlled by the advanced setting *Gratuitous ARP on Fail*.

5.2.4. Host Monitoring for Route Failover

Overview

To provide a more flexible and configurable way to monitor the integrity of routes, CorePlus provides the additional capability to perform *Host Monitoring*. This feature means that one or more external host systems can be routinely polled to check that a particular route is available.

The advantages of Host Monitoring are twofold:

- In a complex network topology it is more reliable to check accessibility to external hosts. Just monitoring a link to a local switch may not indicate a problem in another part of the internal network.
- Host monitoring can be used to help in setting the acceptable Quality of Service level of Internet response times. Internet access may be functioning but it may be desirable to instigate route failover if response latency times become unacceptable using the existing route.

Enabling Host Monitoring

As part of *Route Properties* Host Monitoring can be enabled and a single route can have multiple hosts associated with it for monitoring. Multiple hosts can provide a higher certainty that any network problem resides in the local network rather than because one remote host itself is down.

In association with Host Monitoring there are two numerical parameters for a route:

Grace Period — This is the period of time after startup or after reconfiguration of the Clavister Security Gateway which CorePlus will wait before starting Route Monitoring. This waiting period allows time for all network links to initialize once the security gateway comes online.

Minimum Number of Hosts Available — This is the minimum number of hosts that must be considered to be accessible before the route is deemed to have failed. The criteria for host accessibility are described below.

Specifying Hosts

For each host specified for host monitoring there are a number of property parameters that should be set:

- **Method**
  The method by which the host is to be polled. This can be one of:
  - *ICMP* - ICMP "Ping" polling. An IP address must be specified for this.
  - *TCP* - A TCP connection is established to and then disconnected from the host. An IP address must be specified for this.
  - *HTTP* - A normal HTTP server request using a URL. A URL must be specified for this as well as a text string which is the beginning (or complete) text of a valid response. If no text is specified, any response from the server will be valid.
- **IP Address**
  The IP address of the host when using the **ICMP** or **TCP** option.

- **Port Number**
  The port number for polling when using the **TCP** option.

- **Interval**
  The interval in milliseconds between polling attempts. The default setting is 10,000 and the minimum value allowed is 100 ms.

- **Sample**
  The number of polling attempts used as a sample size for calculating the *Percentage Loss* and the *Average Latency*. This value cannot be less than 1.

- **Maximum Failed Poll Attempts**
  The maximum permissible number of polling attempts that fail. If this number is exceeded then the host is considered unreachable.

- **Max Average Latency**
  The maximum number of milliseconds allowable between a poll request and the response. If this threshold is exceeded then the host is considered unreachable. Average Latency is calculated by averaging the response times from the host. If a polling attempt receives no response then it is not included in the averaging calculation.

The *Reachability Required* option

An important option that can be enabled for a host is the **Reachability Required** option. When this is selected, the host must be determined as accessible in order for that route to be considered to be functioning. Even if other hosts are accessible, this option says that the accessibility of a host with this option set is mandatory.

Where multiple hosts are specified for host monitoring, more than one of them could have **Reachability Required** enabled. If CorePlus determines that any host with this option enabled is not reachable, Route Failover is initiated.

HTTP Parameters

If the **HTTP** polling method is selected then two further parameters can be entered:

- **Request URL**
  The URL which is to be requested.

- **Expected Response**
  The text that is expected back from querying the URL.

  Testing for a specific response text provides the possibility of testing if an application is offline. If, for a instance, a web page response from a server can indicate if a specific database is operational with text such as "*Database OK*", then the absence of that response can indicate that the server is operational but the application is offline.

A Known Issue When No External Route is Specified

With connections to an Internet ISP, an external network route should always be specified. This external route specifies on which interface the network which exists between the Clavister Security Gateway and the ISP can be found. If only an *all-nets* route is specified to the ISP's gateway, route failover may, depending on the connected equipment, not function as expected.

This issue rarely occurs but the reason why it occurs is that ARP queries arriving on a disabled route will be ignored.

5.2.5. Proxy ARP

Overview

As discussed previously in Section 4.4, “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*.

The splitting of an Ethernet network into distinct parts so that traffic between them can be controlled is a common usage of the proxy ARP feature. CorePlus can then be used to monitor and regulate traffic passing between the parts.

A Typical Scenario

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 its 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.

Transparent Mode as an Alternative

Transparent Mode is an alternative and preferred way of splitting ethernet networks. The setup is simpler than using proxy ARP since the administrator need only define the appropriate *switch routes*.

Using switch routes is fully explained in Section 5.7, “Transparent Mode”. In HA clusters, switch routes cannot be used and proxy ARP is the only way to implement transparent mode functionality.

Note

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

5.3. Policy-based Routing

5.3.1. Overview

Policy-based Routing (PBR) is an extension to the standard routing described previously. It offers administrators significant flexibility in implementing routing decision policies by being able to define rules so alternative routing tables are used.

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 routes chosen for traffic can be based on specific traffic parameters.

Policy-based Routing can allow:

Source based routing — A different routing table may need to be chosen based on the source of traffic. 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.

Service-based Routing — A different routing table might need to be chosen based on the service. Policy-based Routing can route a given protocol such as HTTP, through proxies such as Web caches. Specific services might also be routed to a specific ISP so that one ISP handles all HTTP traffic.

User based Routing — A different routing table might need to be chosen based on the user identity or the *group* to which the user belongs. This is particularly useful in *provider-independent metropolitan area networks* where all users share a common active backbone, but each can use different ISPs, subscribing to different providers.

Policy-based Routing implementation in CorePlus is based on two building blocks:

- One or more user-defined alternate *Policy-based Routing Tables* in addition to the standard default **main** routing table.
- One or more *Policy-based routing rules* which determines which routing table to use for which traffic.

5.3.2. 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 sometimes refer to these Policy-based Routing Tables as *alternate* routing tables).

Alternate routing tables contain the same information 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 using alternate tables in conjunction with the **main** table. This is described further in Section 5.3.6, “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.

5.3.3. 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 (HTTP for example) 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.

5.3.4. PBR Table/Interface Association

If a particular routing table is to be always used for traffic from a given source interface, regardless of the service, then there is a way to achieve this.

It is possible to associate the source interface explicitly with a particular table using the **Group** membership parameter for that interface. This is sometimes referred to as *PBR Membership*.

The difference with this method of explicit association is that the administrator cannot specify the service, such as HTTP, for which the lookup will apply. PBR Rules allow a more fine-grained approach to routing table selection by being able to also select a service.

5.3.5. PBR Table Selection

When a packet corresponding to a new connection first arrives, the processing steps are as follows to determine which routing table is chosen:

1. The PBR Rules must first be looked up but to do this the packet's destination interface must be determined and this is always done by a lookup in the *main* routing table. It is therefore important a match for the destination network is found or at least a default **all-nets** route exists which can catch anything not explicitly matched.

2. A search is now made for a Policy-based Routing Rule that matches the packet's source/destination interface/network as well as service. If a matching rule is found then this determines the routing table to use.

3. If no matching PBR rule is found, a check is made to see if the source interface has an alternate routing table explicitly associated with it through the **Group** option. In other words, to see if the interface has *PBR membership* of a particular routing table. If there is no membership then the *main* table will be used.

4. Once the correct routing table has been located, a check is made to make sure that the source IP address in fact belongs on the receiving interface. The Access Rules are firstly examined to see if they can provide this check (see Section 7.1, “Access Rules” for more details of this feature). If there are no Access Rules or a match with the rules cannot be found, a reverse lookup in the previously selected routing table is done using the source IP address. If the check fails then a **Default access rule** log error message is generated.

5. At this point, using the routing table selected, the actual route lookup is done to find the packet's destination interface. At this point the *ordering* parameter is used to determine how the actual lookup is done and the options for this are described in the next section. To implement virtual systems, the *Only* ordering option should be used.

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. Note that the original route lookup to find the destination interface used for all rule look-ups was done with the original, untranslated address.

7. If allowed by the IP rule set, the new connection is opened in the CorePlus state table and the packet forwarded through this connection.

5.3.6. 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 available options are:

1. **Default** - The default behavior is to first look up the route in the **main** table. If no matching route is found, or the default route is found (the route with the destination **all-nets** - 0.0.0.0/0), a lookup for a matching route in the alternate table is done. If no match is found in the alternate table then the default route in the main table will be used.

2. **First** - This behavior is to first look up the connection's route in the alternate table. If no matching route is found there then the **main** table is used for the lookup. The default **all-nets** route will be counted as a match in the alternate table if it exists there.

3. **Only** - This option ignores the existence of any other table except the alternate table so the alternate table is the only one used for the lookup. One application of this is to give the administrator a way to dedicate a single routing table to one set of interfaces. **Only** is the option to use when creating virtual systems since it can dedicate one routing table to a 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 the default route with a destination interface of **all-nets** in the default main routing table. If there is no route that is an exact match then the absence of a default **all-nets** route will mean that the connection will be dropped. The alternative of having interfaces associated explicitly with alternate routing tables through the **Group** option will also not function if the default route is missing.*

Example 5.3. Creating a Policy-based Routing Table

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

Web Interface

1. Go to **Routing > Routing Tables > Add > RoutingTable**
2. Now enter:
   - **Name:** TestPBRTable
   - 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 (*all-nets*), 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.
3. If **Remove Interface IP Routes** is enabled, the default interface routes are removed, that is to say routes to the *core* interface (which are routes to CorePlus itself).
4. Click **OK**

Example 5.4. Creating the Route

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

Web Interface

1. Go to **Routing > Routing Tables > TestPBRTable > Add > Route**
2. 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)
3. Click **OK**

Example 5.5. 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 10.10.10.0/24 belonging to "ISP A" and "20.20.20.0/24" belonging to "ISP B". The ISP gateways are 10.10.10.1 and 20.20.20.1, 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 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	10.10.10.0/24		wan1
lan1	20.20.20.0/24		wan2
wan1	10.10.10.1/32		lan1
wan2	20.20.20.1/32		lan1
wan1	all-nets	10.10.10.1	

Contents of the named Policy-based Routing table r2:

Interface	Network	Gateway
wan2	all-nets	20.20.20.1

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 (*all-nets*).

Contents of the Policy-based Routing Policy:

Source Interface	Source Range	Destination Interface	Destination Range	Service	Forward VR table	Return VR table
lan1	10.10.10.0/24	wan2	all-nets	ALL	r2	r2
wan2	all-nets	lan1	20.20.20.0/24	ALL	r2	r2

To configure this example scenario:

Web Interface

1. Add the routes found in the list of routes in the main routing table, as shown earlier.
2. Create a routing table called "r2" and make sure the ordering is set to "Default".
3. Add the route found in the list of routes in the routing table "r2", as shown earlier.
4. Add two VR policies according to the list of policies shown earlier.
   - Go to **Routing > Routing Rules > Add > Routing Rule**
   - 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.

5.4. Virtual Routing

5.4.1. Overview

Virtual Routing is a CorePlus feature that allows the creation of multiple, logically separated *virtual systems*, each with their own routing table. These virtual systems can behave as physically separated Clavister Security Gateways and nearly everything that can be done with separate gateways can be done with them, including running dynamic routing processes such as OSPF. Virtual systems are sometimes referred to as *Virtual Routers*.

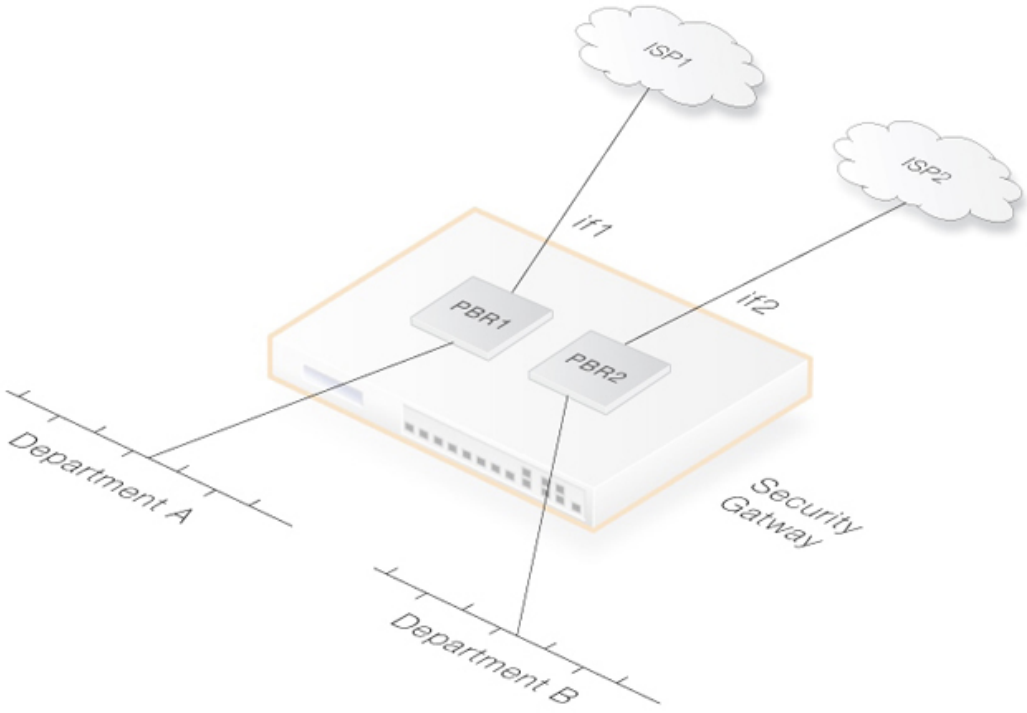
The CorePlus components involved in virtual routing are:

- Separate Policy-based routing tables for each virtual system.
- Per-interface policy-based routing table membership to make interface IP addresses reachable only using a particular routing table. This association can also be made by explicitly linking an interface with a specific routing table.
- Pairs of loopback interfaces for communication between virtual systems if required.

Virtual Routing is different from using *Policy-based Routing Rules* where a rule is used to define which routing table should be used. In virtual routing the table is chosen based only on the interface from which the traffic enters. The interface might be physical or it might be a virtual lan (VLAN).

5.4.2. A Simple Scenario

Consider a single Clavister Security Gateway connected to two external ISPs, *ISP1* and *ISP2*. *ISP1* services department C and connects via interface **if1**. *ISP2* services department D and connects via interface **if2**. For administration purposes, it is decided that the Clavister Security Gateway is to be broken into 2 virtual systems, one for each ISP.



This is done by creating two policy-based routing tables, *PBR1* to deal with *ISP1* to department C

traffic and another, *PBR2*, to deal with *ISP2* to department D traffic. *if1* will be a member of the first table but not the second. *if2* will be a member of the second table but not the first. It is this interface membership that determines which routing table will be used and which therefore keeps traffic totally separated. The *main* routing table could be used in this example as one of the two routing tables so that only one additional routing table needs to be created instead of two.

Associating an interface with a routing table is, as mentioned above, not necessarily done only through interface membership. CorePlus allows routing tables to be explicitly associated with a particular interface so that all traffic received on that interface is subject to one specific routing table.

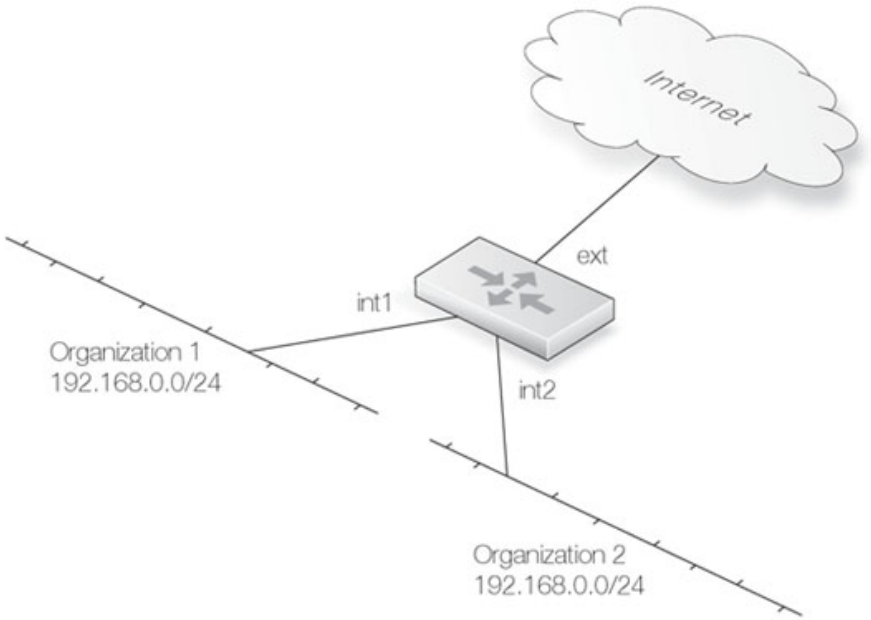
Additionally, VPN tunnels could be set up from the outside to the Clavister Security Gateway using the interfaces as the tunnel endpoints. The tunnels may be then separated logically so they use different routing tables.

In this simple example *loopback interfaces* were not used since there is no requirement for communication between the virtual systems. For example, department C does not need to communicate with department D. If inter-system communication is needed then an appropriate loopback interface pair would have to be defined to allow this.

One advantage of using separate routing tables on different interfaces is that IP address ranges could potentially be re-used on each interface.

5.4.3. Advantages Over PBR Rules

Using Policy-based Routing Rules can solve many of the problems that virtual routing can but complex configurations can become unwieldy for PBR rules. Consider a single Clavister Security Gateway being used as a firewall for two organizations, both using the same IP span.



In this case two routing tables could be used, one for each organization:

pbrtable1 routing table

	Interface	Network	Gateway
1	ext	all-nets	defaultgw
2	int1	192.168.0.0/24	

pbrtable2 routing table

	Interface	Network	Gateway
1	ext	all-nets	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, outbound as well as inbound traffic is handled with four rules:

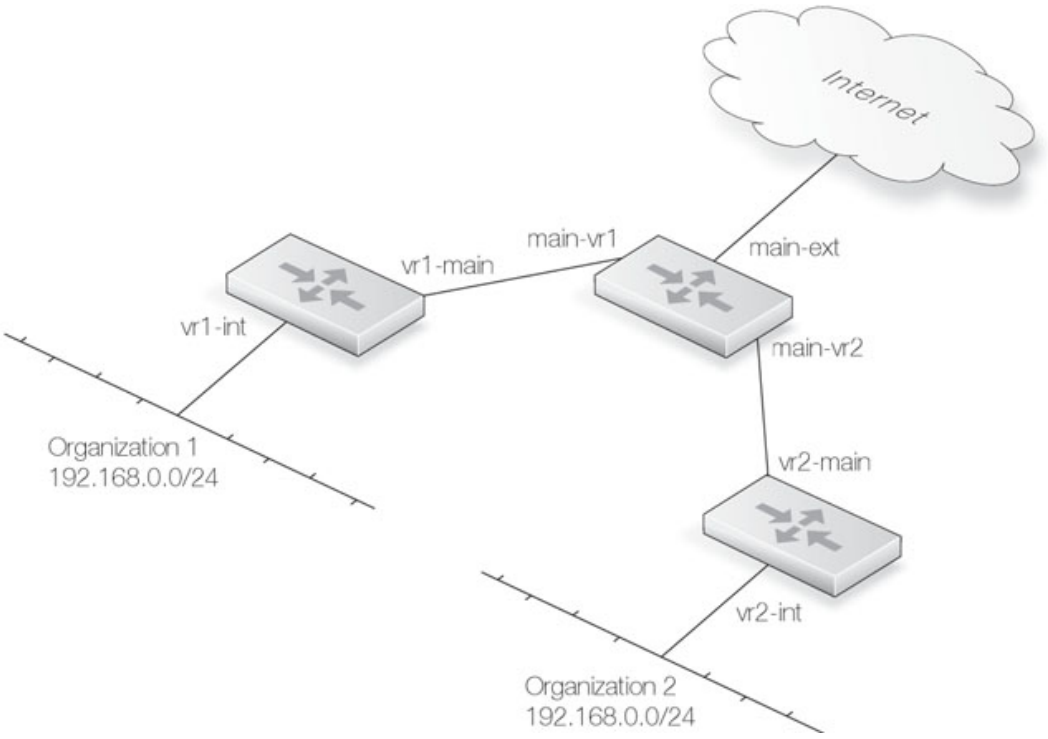
	Name	Source if	Source Net	Dest Net	Fwd PBR	Ret PBR
1	org1-in	ext	all-nets	pubip-org1	pbrtable1	pbrtable1
2	org1-out	int1	all-nets	all-nets	pbrtable1	pbrtable1
3	org2-in	ext	all-nets	pubip-org2	pbrtable2	pbrtable2
4	org2-out	int2	all-nets	all-nets	pbrtable2	pbrtable2

This works if the two organizations do not attempt to communicate with each other. If this happens the following two PBR rules are also needed, before the other four.

	Name	Source if	Source Net	Dest Net	Fwd PBR	Ret PBR
1	org1-org2	int1	all-nets	pubip-org2	pbrtable1	pbrtable2
2	org2-org1	int2	all-nets	pubip-org1	pbrtable2	pbrtable1

With two organizations, two PBR rules are enough for them to communicate. However, with three organizations, six are needed; with four, twelve are needed; with five, twenty are needed. The numbers continue: 30, 42, 56... and the administration task becomes unmanageable.

Virtual Routing can eliminate the all-to-all mappings described in the previous section. Using interface PBR table membership instead of PBR rules reduces the complexity by creating 3 virtual systems as shown above.



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

main routing table

	Interface	Network	Gateway
1	main-ext	all-nets	defaultgw
2	main-vs1	pubip-vs1	
3	main-vs2	pubip-vs2	

vs1 routing table

	Interface	Network	Gateway
1	vs1-main	all-nets	
2	vs1-int	192.168.0.0/24	

vs2 routing table

	Interface	Network	Gateway
1	vs2-main	all-nets	
2	vs2-int	192.168.0.0/24	

Ethernet Interfaces

	Name	IP address	PBR table
1	main-ext	ip_main-ext	main
2	vs1-int	192.168.0.1	vs1
3	vs2-int	192.168.0.254	vs2

Loopback Interfaces

	Interface	IP address	Loop to	PBR table
1	main-vs1	ip_main-ext	vs1-main	main
2	vs1-main	pubip-vs1	main-vs1	vs1
3	main-vs2	ip_main-ext	vs2-main	main
4	vs2-main	pubip-vs2	main-vs2	vs2

For each connection between a pair of virtual systems, two loopback interfaces are used, one for each system. When traffic is sent through "main-vs1", it arrives on "vs1-main". When traffic is sent through "vs1-main", it is received on "main-vs1". This is exactly the same as with two Clavister Security Gateways: two interfaces, one on 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 note how the IP addresses of the internal interfaces of the virtual systems 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 "vs2" even though it should not be, and 192.168.0.254 being unusable in "vs1". With per-interface routing table membership, interface IP addresses belonging to one virtual system will not interfere with other systems and loopback interfaces are not needed.

The IP addresses of the "main-vs1" and "main-vs2" 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 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 the issue exists of traffic originating from the Clavister Security 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.

5.4.4. IP Rules with Virtual Routing

The IP rule sets for different virtual systems can be split into separate rule sets using the CorePlus feature of creating multiple IP rule sets (see Section 4.5.4, “Multiple IP Rule Sets” for more detail on this feature).

IP Rules for different virtual systems need not be split up. Instead, the rules can reside together in the standard IP rule set. The benefit is being able to define "shared" or "global" rules that span over several virtual systems. For instance, if an aggressive worm attacks, it may be desirable to drop all communication on ports known to be used by the worm until counter-measures can be put into place. One single *Drop* rule placed at the top of the rule set can take care of this for all the virtual systems. Using the example of the previous section, this is how the IP rule set might look:

Interface Groups

	Name	Interfaces
1	main-vsifs	main-vs1, main-vs2
2	main-ifs	main-ext, main-vsifs
3	vs1-ifs	vs1-main, vs1-int
4	vs2-ifs	vs2-main, vs2-int

IP Rules

	Name	Action	Source if	Source Net	Dest if	Dest Net	Service
IP Rules for the "main" virtual system							
1	main-allowall	Allow	main-ifs	all-nets	Any	all-nets	All
IP Rules for the "vs1" virtual system							
2	vs1-http-in	SAT	vs1-ifs	all-nets	pubip-vs1	all-nets	http, SetDest 192.168.0.5
3	vs1-http-in	Allow	vs1-main	all-nets	Any	pubip-vs1	http
4	vs1-out	NAT	vs1-int	all-nets	Any	all-nets	All
IP Rules for the "vs2" virtual system							
5	vs2-smtp-in	SAT	vs2-main	all-nets	Any	pubip-vs2	All
6	vs2-smtp-in	Allow	vs2-main	all-nets	Any	pubip-vs2	smtp
7	vs2-http-out	NAT	vs2-int	192.168.0.4	vs2-main	all-nets	http

Note that 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" routing table. 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. Such problems are eliminated with virtual routing.

The source interface filters are very specific. "Any" is not used as the source interface anywhere, since such a rule would trigger regardless. Consider for instance what would happen if the "vs1-http-in" rules were to use "Any" as source interface. They would trigger as soon as packets destined to "pubip-vs1" 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 Clavister Security Gateway.

If the same naming scheme as shown in this example is used, making sure the source interfaces are correct can be done quickly. All the rules concerning the "main" system have source interfaces beginning with "main-". All those concerning "vs1" have source interfaces beginning with "vs1-", and so on.

The destination interface filters, however, do not need to be as specific as the source interface filters. The possible destinations are limited by the routing tables used. If the "vs1" table only includes routes through "vs1-" interfaces, "Any" filters can only mean "through other interfaces in the same virtual system". It may however be sound practice to write tighter destination interface filters in case an error occurs elsewhere in the configuration. In this example, rule 1 might use "main-ifs", rule 4 might use "vs1-main". The SAT and corresponding Allow rules however are already fairly tight in that they only concern one single destination IP address.

5.4.5. Multiple IP rule sets

An alternative approach to having all the IP rules for different virtual systems in one rule set is to make use of *Multiple IP rule sets*.

Although all scanning of IP rules begins in the *main* rule set, it is possible to define a rule in *main* whose action is **Goto** so that scanning continues in a separate, named rule set. These extra rule sets can be defined as needed and one rule set can be created for each virtual system and its corresponding routing table.

More details on this subject can be found in Section 4.5.4, “Multiple IP Rule Sets”.

5.4.6. 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 <pbrtable>" to source pings from different virtual systems
- Use "ping -r <recvif> -s <srcip>" to test the rule set, 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 <pbrtable> -all" to view all route entries in a given table, including "core" routes
- Use "route -lookup <ipaddr> <pbrtable>" to make sure that a given IP address is routed the way you think in a given virtual system. (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 system, a separate rule decision is made, and a separate connection is established

5.5. Dynamic Routing

5.5.1. 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.

The *Distance Vector* Algorithm

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 determines 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.

Link State Algorithms

In contrast to 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

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.

Comparing Dynamic Routing Algorithms

Due to the fact that the global link state information is maintained everywhere in a network, LS algorithms offer a high degree of configuration control and scalability. Changes result in broadcasts

of just the updated information to other routers which means 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 many advantages it offers.

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:

Path length 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.

Item Bandwidth The traffic capacity of a path, rated by "Mbps".

Load The usage of a router. The usage can be evaluated by CPU utilization and throughput.

Delay 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.

5.5.2. OSPF

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 sub netted 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.

OSPF Areas

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

Areas 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.

ABRs 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.

ASBRs 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.

Backbone Areas 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.

Stub Areas 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.

Transit Areas Transit areas are used to pass traffic from an area that is not directly connected to the backbone area.

The 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.

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:

Down This is the initial state of the neighbor relationship.

Init 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 *2-way* state.

2-Way 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 *Full*. On Broadcast interfaces, only the DR/BDR will advance to *Full*state with their neighbors, all the remaining neighbors will remain in the 2-Way state.

ExStart Preparing to build adjacency.

Exchange Routers are exchanging Data Descriptors.

Loading Routers are exchanging LSAs.

Full This is the normal state of an adjacency between a router and the DR/BDR.

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.

Virtual Links

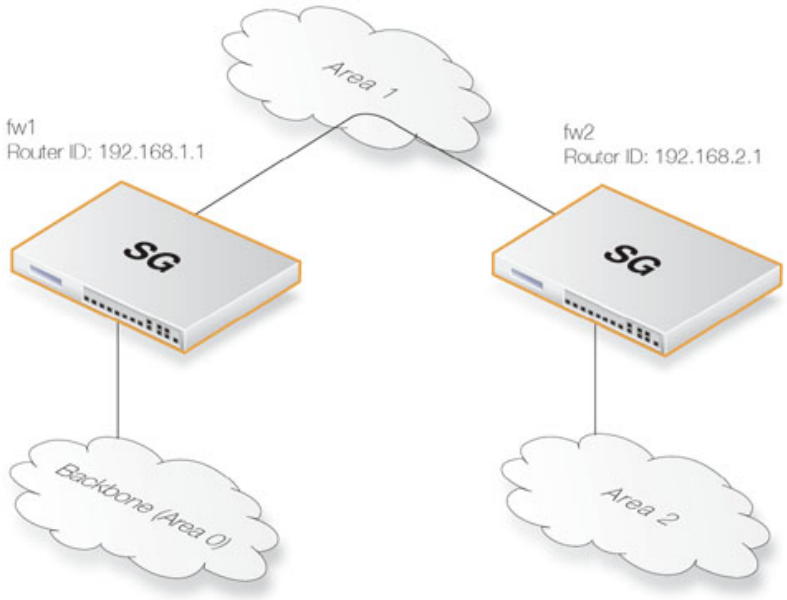
Virtual links are used for:

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

Areas 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 5.2. Virtual Links Example 1

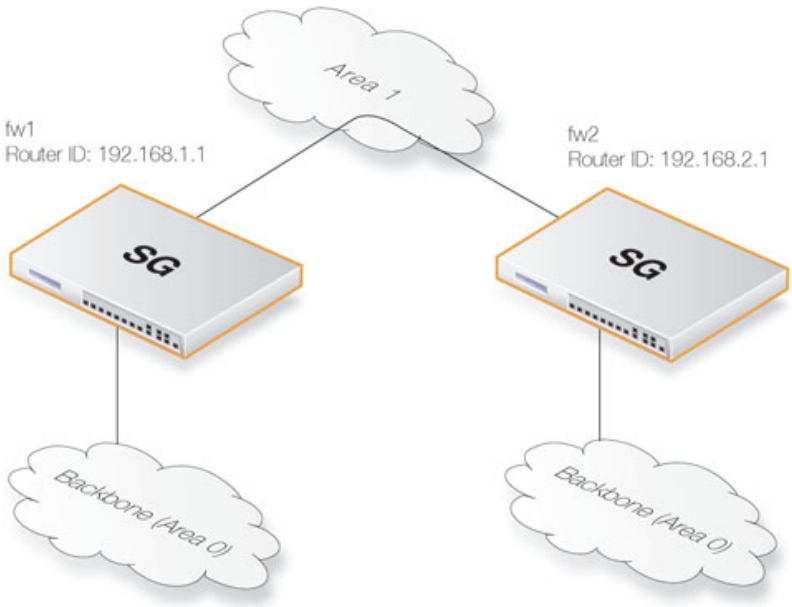


In the above example, the Virtual Link is configured between fw1 and fw2 on Area 1, as it is used as the transit area. In this configuration only the Router ID has to be configured. The diagram shows that fw2 needs to have a Virtual Link to fw1 with Router ID 192.168.1.1 and vice versa. These Virtual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

A 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 from each side and have a common area in between.

Figure 5.3. 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.

OSPF High Availability Support

There are some limitations in High Availability 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 Clavister Security Gateway needs to have a broadcast interface with at least ONE neighbor for ALL areas that the security 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 an HA cluster on the same broadcast network without any other neighbors (they will not form adjacency with each other because of the router priority 0). However, it may be possible, depending on the scenario, to setup a point to point link between them instead. Special care must also be taken when setting up a virtual link to an security gateway in an HA cluster. The endpoint setting up a link to the HA security gateway must setup 3 separate links: one to the shared, one to the master and one to the slave router id of the security gateway.

5.5.3. Dynamic Routing Policy

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 on. 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.

Example 5.6. 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. 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 (which is the same as specifying the IP address *0.0.0.0/0*) are allowed.

CLI

```
Device:/> add DynamicRoutingRule OSPFProcess=as0 Name=ImportOSPFRoutes
     DestinationNetworkExactly=all-nets
```

Web Interface

1. Go to **Routing > Dynamic Routing Rules > Add > Dynamic routing policy rule**
2. Specify a suitable name for the filter. For example, *ImportOSPFRoutes*.
3. In the **Select OSPF Process**, select *as0*
4. Choose **all-nets** in the **Exactly Matches** dropdown control
5. 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*.

CLI

```
Device:/> cc DynamicRoutingRule ImportOSPFRoutes

Device:/ImportOSPFRoutes> add DynamicRoutingRuleAddRoute
                          Destination=MainRoutingTable
```

Web Interface

1. Go to **Routing > Dynamic Routing Rules**
2. Click on the recently created **ImportOSPFRoutes**
3. Go to **OSPF Routing Action > Add > DynamicRountingRuleAddRoute**
4. In **Destination**, add the main routing table to the **Selected** list
5. Click **OK**

Example 5.7. 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. First, add a dynamic routing policy filter that matches the main routing table and the default route:

CLI

```
Device:/> add DynamicRoutingRule OSPFProcess=as0 name=ExportDefRoute
                RoutingTable=MainRoutingTable DestinationInterface=wan
                DestinationNetworkExactly=all-nets
```

Web Interface

1. Go to **Routing > Dynamic Routing Rules > Add > Dynamic routing policy rule**
2. Specify a suitable name for the filter, for example *ExportDefRoute*
3. For **From Routing Table** select **Main Routing Table**
4. Choose **wan** for **Destination Interface**
5. Choose **all-nets** in the **...Exactly Matches** list
6. Click **OK**

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

CLI

```
Device:/> cc DynamicRoutingRule ExportDefRoute

Device:/ExportDefRoute/> add DynamicRoutingRuleExportOSPF ExportToProcess=as0
```

Web Interface

1. Go to **Routing > Dynamic Routing Rules**
2. Click on the newly created **ExportDefRoute**
3. Go to **OSPF Action > Add > DynamicRoutingRuleExportOSPF**
4. For **Export to process** choose **as0**
5. Click **OK**

5.6. Multicast Routing

5.6.1. 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 (in other words, to CorePlus itself). *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 4.3.2, “Ethernet Interfaces”.*

5.6.2. 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 may 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:

Using 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 behavior of CorePlus.

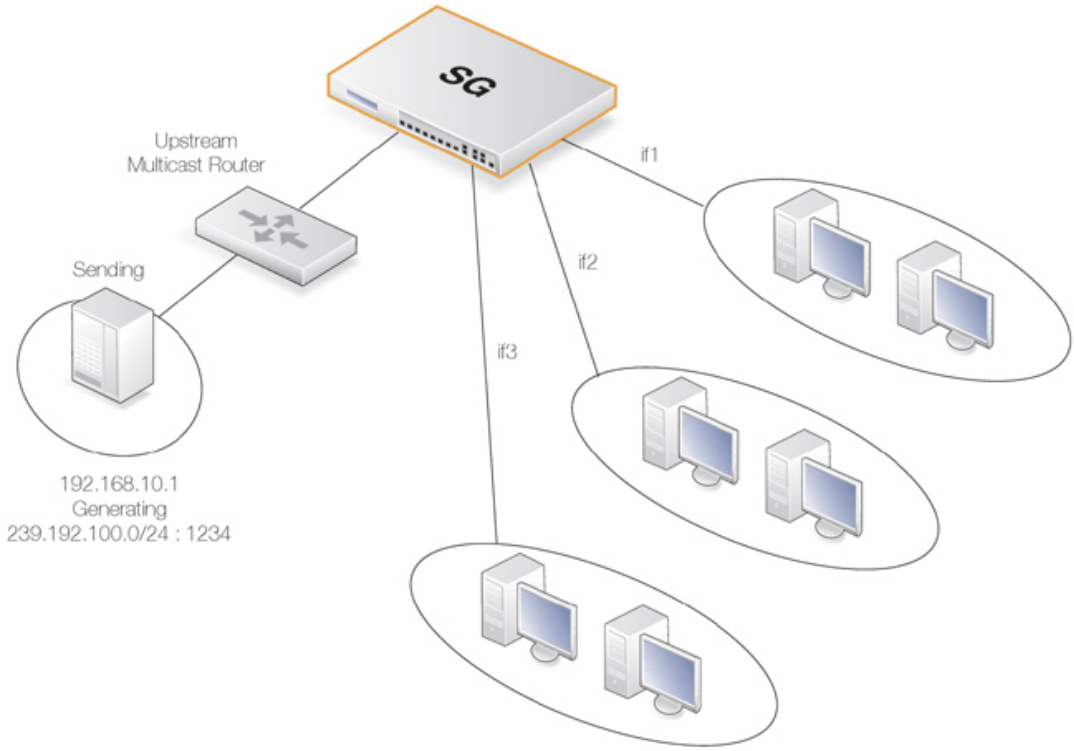
Not using IGMP The traffic flow will be forwarded according to the specified interfaces directly without any inference from IGMP.

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

5.6.2.1. 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 5.6.3.1, “IGMP Rules Configuration - No Address Translation”.

Figure 5.4. Multicast Forwarding - No Address Translation



Note: SAT Multiplex rules must have a matching Allow rule
Remember to add an Allow rule matching the SAT Multiplex rule.

Example 5.8. 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.

Web Interface

A. Create a custom service for multicast called *multicast_service*:

1. Go to **Objects > Services > Add > TCP/UDP**
2. Now enter:
   - **Name:** multicast_service
   - **Type:** UDP
   - **Destination:** 1234

B. Create an IP rule:

1. Go to **Rules > IP Rules > Add > IP Rule**
2. Under **General** enter.
   - **Name:** a name for the rule, for example *Multicast_Multiplex*
   - **Action:** Multiplex SAT
   - **Service:** multicast_service
3. Under **Address Filter** enter:
   - **Source Interface:** wan
   - **Source Network:** 192.168.10.1
   - **Destination Interface:** core
   - **Destination Network:** 239.192.10.0/24
4. Click the **Multiplex SAT** tab 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.
5. Make sure the **forwarded using IGMP** checkbox is set
6. Click **OK**

Creating Multiplex Rules with the CLI

Creating multiplex rules through the CLI requires some additional explanation.

First, the *IPRuleset*, in this example *main*, needs to be selected as the current category:

```
Device:/> cc IPRuleset main
```

The CLI command to create the multiplex rule is then:

```
add IPRule SourceNetwork=<srcnet> SourceInterface=<srcif>
        DestinationInterface=<srcif> DestinationNetwork=<destnet>
        Action=MultiplexSAT Service=<service>
        MultiplexArgument={outif1;ip1},{outif2;ip2},{outif3;ip3}...
```

The two values *{outif;ip}* represent a combination of output interface and, if address translation of a group is needed, an IP address.

If, for example, multiplexing of the multicast group *239.192.100.50* is required to the output interfaces *if2* and *if3*, then the command to create the rule would be:

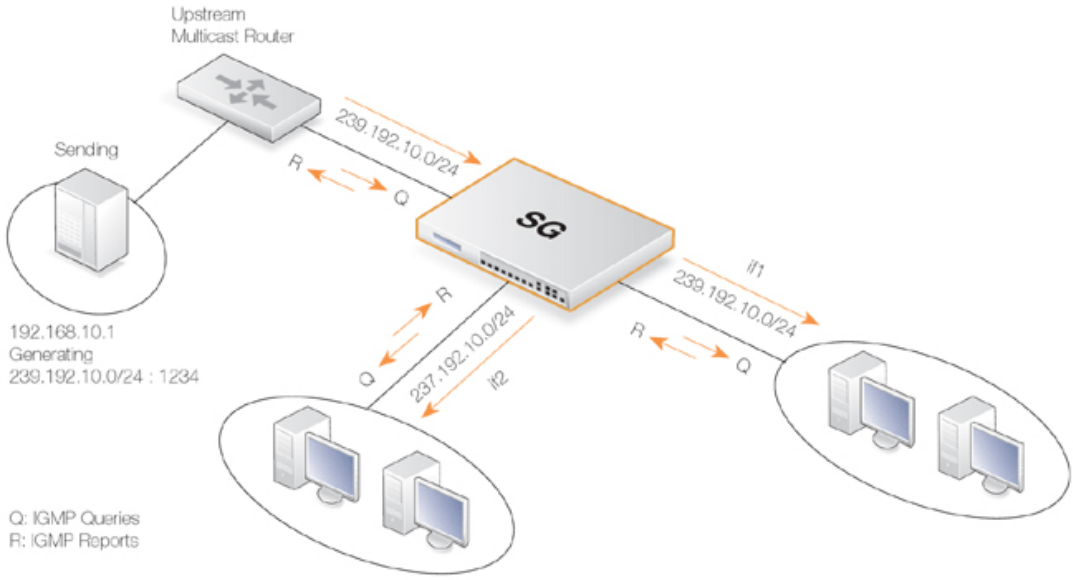
```
add IPRule SourceNetwork=<srcnet> SourceInterface=<if1>
        DestinationInterface=core DestinationNetwork=239.192.100.50
        Action=MultiplexSAT Service=<service>
        MultiplexArgument={if2;},{if3;}
```

The destination interface is *core* since *239.192.100.50* is a multicast group. No address translation of *239.192.100.50* was added but if it is required for, say, *if2* then the final argument would be:

```
        MultiplexArgument={if2;<new_ip_address>},{if3;}
```

5.6.2.2. Multicast Forwarding - Address Translation Scenario

Figure 5.5. 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 5.6.3.2, “IGMP Rules Configuration - Address Translation”.

Caution

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

Example 5.9. Multicast Forwarding - Address Translation

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

Web Interface

A. Create a custom service for multicast called *multicast_service*:

1. Go to **Objects > Services > Add > TCP/UDP**
2. Now enter:
   - **Name:** multicast_service
   - **Type:** UDP
   - **Destination:** 1234

B. Create an IP rule:

1. Go to **Rules > IP Rules > Add > IP Rule**
2. Under **General** enter.
   - **Name:** a name for the rule, for example *Multicast_Multiplex*
   - **Action:** Multiplex SAT
   - **Service:** multicast_service
3. Under **Address Filter** enter:
   - **Source Interface:** wan
   - **Source Network:** 192.168.10.1
   - **Destination Interface:** core
   - **Destination Network:** 239.192.10.0/24
4. Click the **Multiplex SAT** tab
5. Add interface **if1** but leave the **IPAddress** empty
6. Add interface **if2** but this time, enter *237.192.10.0* as the **IPAddress**
7. Make sure the **Forwarded using IGMP** checkbox is enabled
8. Click **OK**

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

5.6.3. 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 neighboring router 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 5.6. Multicast Snoop

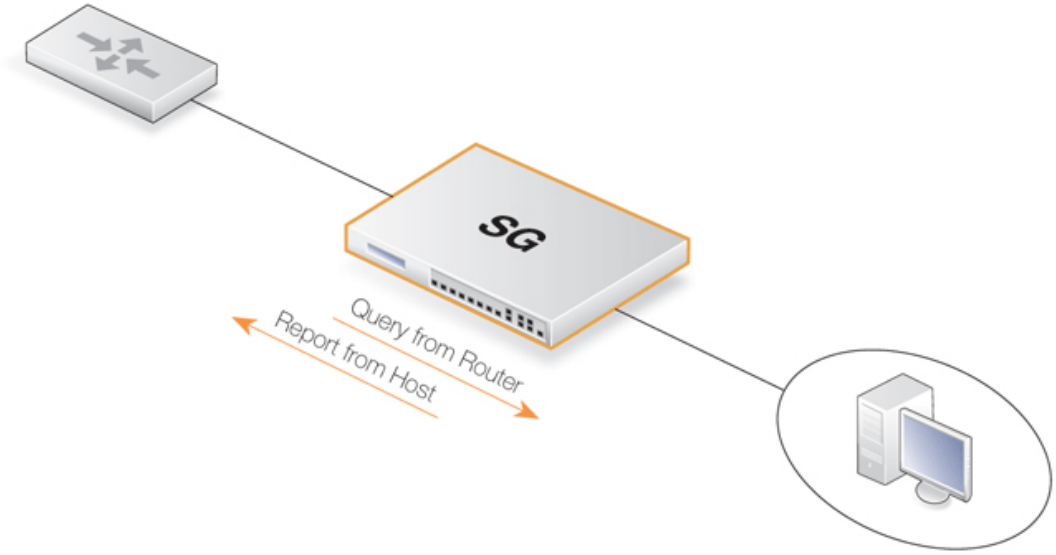
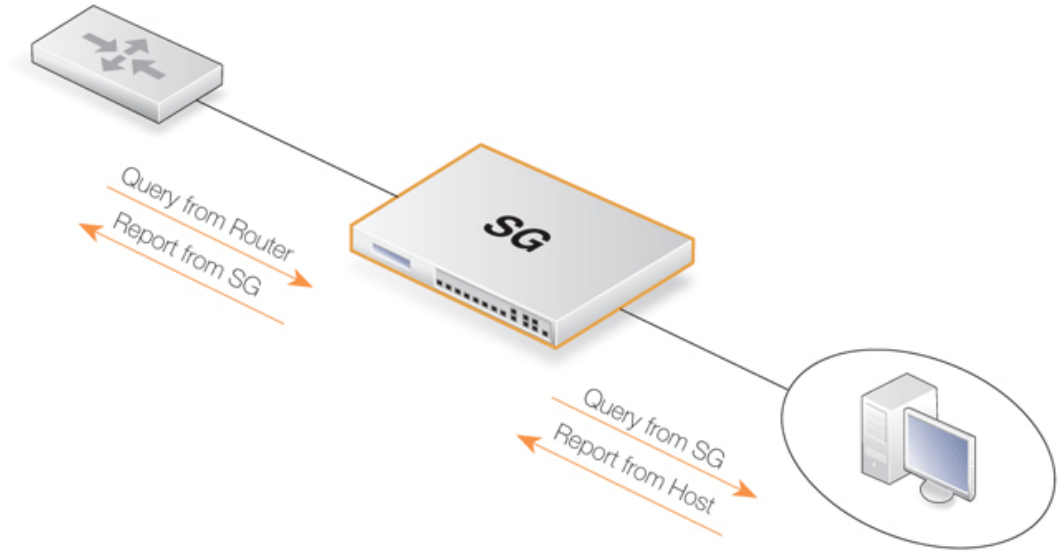


Figure 5.7. Multicast Proxy



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.

5.6.3.1. 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 5.10. 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.

Web Interface

A. Create the first IGMP Rule:

1. Go to **Routing > IGMP > IGMP Rules > Add > IGMP Rule**
2. Under **General** enter:
   - **Name:** A suitable name for the rule, for example *Reports*
   - **Type:** Report
   - **Action:** Proxy
   - **Output:** wan *(this is the relay interface)*
3. Under **Address Filter** enter:
   - **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
4. Click **OK**

B. Create the second IGMP Rule:

1. Again go to **Routing > IGMP > IGMP Rules > Add > IGMP Rule**
2. Under **General** enter:
   - **Name:** A suitable name for the rule, for example *Queries*
   - **Type:** Query
   - **Action:** Proxy
   - **Output:** IfGrpClients *(this is the relay interface)*
3. Under **Address Filter** enter:
   - **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
4. Click **OK**

5.6.3.2. 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 5.6.2.2, “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 5.11. *if1* Configuration

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

Web Interface

A. Create the first IGMP Rule:

1. Go to **Routing > IGMP > IGMP Rules > Add > IGMP Rule**
2. Under **General** enter:
   - **Name:** A suitable name for the rule, for example *Reports_if1*
   - **Type:** Report
   - **Action:** Proxy
   - **Output:** wan *(this is the relay interface)*
3. Under **Address Filter** enter:
   - **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
4. Click **OK**

B. Create the second IGMP Rule:

1. Again go to **Routing > IGMP > IGMP Rules > Add > IGMP Rule**
2. Under **General** enter:
   - **Name:** A suitable name for the rule, for example *Queries_if1*
   - **Type:** Query
   - **Action:** Proxy
   - **Output:** if1 *(this is the relay interface)*
3. Under **Address Filter** enter:
   - **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

4. Click **OK**

Example 5.12. *if2* Configuration - Group Translation

The following steps needs 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.

Web Interface

A. Create the first IGMP Rule:

1. Go to **Routing > IGMP > IGMP Rules > Add > IGMP Rule**
2. Under **General** enter:
   - **Name:** A suitable name for the rule, for example *Reports_if2*
   - **Type:** Report
   - **Action:** Proxy
   - **Output:** wan *(this is the relay interface)*
3. Under **Address Filter** enter:
   - **Source Interface:** if2
   - **Source Network:** if2net
   - **Destination Interface:** core
   - **Destination Network:** auto
   - **Multicast Source:** 192.168.10.1
   - **Multicast Group:** 239.192.10.0/24
4. Click **OK**

B. Create the second IGMP Rule:

1. Again go to **Routing > IGMP > IGMP Rules > Add > IGMP Rule**
2. Under **General** enter:
   - **Name:** A suitable name for the rule, for example *Queries_if2*
   - **Type:** Query
   - **Action:** Proxy
   - **Output:** if2 *(this is the relay interface)*
3. Under **Address Filter** enter:
   - **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
4. Click **OK**

Advanced IGMP Settings

There are a number of IGMP advanced settings which are global and apply to all interfaces which do not have IGMP settings explicitly specified for them.

5.6.4. Advanced IGMP Settings

Auto Add Multicast Core Route

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*

IGMP Before Rules

For IGMP traffic, by-pass the normal IP rule set and consult the IGMP rule set.

Default: *Enabled*

IGMP React To Own Queries

The security gataway should always respond with IGMP Membership Reports, even to queries orginating from itself. Global setting on interfaces without an overriding IGMP Setting.

Default: *Disabled*

IGMP Lowest Compatible Version

IGMP messages with a version lower than this will be logged and ignored. Global setting on interfaces without an overriding IGMP Setting.

Default: *IGMPv1*

IGMP Router Version

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*

IGMP Last Member Query Interval

The maximum time in milliseconds 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*

IGMP Max Total Requests

The maximum global number of IGMP messages to process each second.

Default: *1000*

IGMP Max Interface Requests

The maximum number of requests per interface and second. Global setting on interfaces without an overriding IGMP Setting.

Default: *100*

IGMP Query Interval

The interval in milliseconds between General Queries sent by the device to refresh its IGMP state. Global setting on interfaces without an overriding IGMP Setting.

Default: *125,000*

IGMP Query Response Interval

The maximum time in milliseconds until a host has to send a reply to a query. Global setting on interfaces without an overriding IGMP Setting.

Default: *10,000*

IGMP Robustness Variable

IGMP is robust to (IGMP Robustness Variable - 1) packet losses. Global setting on interfaces without an overriding IGMP Setting.

Default: *2*

IGMP Startup Query Count

The security gateway will send IGMP Startup Query Count general queries with an interval of IGMPStartupQueryInterval at startup. Global setting on interfaces without an overriding IGMP Setting.

Default: *2*

IGMP Startup Query Interval

The interval of General Queries in milliseconds used during the startup phase. Global setting on interfaces without an overriding IGMP Setting.

Default: *30,000*

IGMP Unsolicated Report Interval

The time in milliseconds between repititions of an initial membership report. Global setting on interfaces without an overriding IGMP Setting.

Default: *1,000*

5.7. Transparent Mode

5.7.1. Overview

Deploying Transparent Mode

The CorePlus *Transparent Mode* feature allows a Clavister Security Gateway to be placed at a point in a network without any reconfiguration of the network and without hosts being aware of its presence. All CorePlus features can then be used to monitor and manage traffic flowing through that point. CorePlus can allow or deny access to different types of services (for example HTTP) and in specified directions. As long as users are accessing the services permitted, they will not be aware of the Clavister Security Gateway's presence.

Network security and control can therefore be significantly enhanced with deployment of a Clavister Security Gateway operating in Transparent Mode but while disturbance to existing users and hosts is minimized.

Switch Routes

Transparent Mode is enabled by specifying a *Switch Route* instead of a standard *Route* in routing tables. The switch route usually specifies that the network *all-nets* is found on a specific interface. CorePlus then uses ARP message exchanges over the connected Ethernet network to identify and keep track of which host IP addresses are located on that interface (this is explained further below). There should not be a normal non-switch route for that same interface.

In certain, less usual circumstances, switch routes can have a network range specified instead of *all-nets*. This is usually when a network is split between two interfaces but the the administrator does not know exactly which users are on which interface.

Usage Scenarios

Two examples of Transparent Mode's usage are:

- **Implementing Security Between Users**

  In a corporate environment, there may be a need to protect the computing resources of different departments from one another. The finance department might require access to only a restricted set of services (HTTP for example) 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 hosts. By deploying a single Clavister Security Gateway between the two department's physical networks, transparent but controlled access can be achieved.

- **Controlling Internet Access**

  An organization allows traffic between the external Internet and a range of public IP addresses 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. This usage is dealt with in greater depth below in Section 5.7.2, “Enabling Internet Access”.

Comparison with Routing Mode

The Clavister Security Gateway can operate in two modes: *Routing Mode* using non-switch routes or *Transparent Mode* using switch routes. With non-switch routes, the Clavister Security Gateway performs all the functions of an *OSI Layer 3 Router*. If the Clavister Security Gateway is placed into a network for the first time, or if network topology changes, the routing configuration must therefore

be checked and adjusted 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 works well when comprehensive control over routing is desired.

With **switch routes**, the Clavister Security Gateway operates in Transparent Mode and resembles a *OSI Layer 2 Switch*: it screens IP packets and forwards them transparently to the correct interface without modifying any of the source or destination information at the IP or Ethernet levels. This is done by CorePlus keeping track of the MAC addresses of the connected hosts and CorePlus allows physical Ethernet networks on either side of the Clavister Security Gateway to act as though they were a single logical IP network. (See Appendix D, *The OSI Framework* for an overview of the OSI layer model.)

Two benefits of Transparent Mode over conventional routing are:

- A user can move from one interface to another in a "plug-n-play" fashion, without changing their IP address (assuming their IP address is fixed). The user can still obtain the same services as before (for example HTTP, FTP) without any need to change routes.
- The same network address range can exist on several interfaces.

Note: Transparent and Routing Mode can be combined

Transparent Mode and Routing Mode can operate together on a single Clavister Security Gateway. Switch Routes can be defined alongide standard non-switch routes although the two types cannot be combined for the same interface. An interface operates in one mode or the other.

It is also possible to create a hybrid case by applying address translation on otherwise transparent traffic.

How Transparent Mode Works

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 Clavister Security Gateway.

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 Clavister Security 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 security 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.

Enabling Transparent Mode

The following steps are required to enable CorePlus Transparent Mode:

1. The interfaces that are to be transparent should be first collected together into a single *Interface Group* object. Interfaces in the group should be marked as **Security transport equivalent** if hosts are to move freely between them.

2. A **Switch Route** is now created in the appropriate routing table and the interface group associated with it. Any existing non-switch routes for interfaces in the group should be removed from the routing table.

   For the **Network** parameter in the switch route, specify *all-nets* or alternatively, specify the range of IP addresses that will be transparent between the interfaces.

3. Create the appropriate IP rules ín the IP rule set to allow the desired traffic to flow between the interfaces operating in Transparent Mode.

   If no restriction at all is to be initially placed on traffic flowing in transparent mode, the following single IP rule could be added but more restrictive IP rules are recommended.

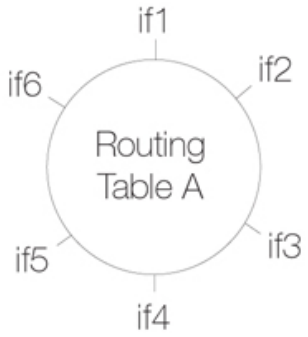
Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
Allow	any	all-nets	any	all-nets	all

Multiple Switch Routes are Connected Together

The setup steps listed above describe placing all the interfaces into a single interface group object which is associated with a single switch route.

An alternative to one switch route is to not use an interface group but instead use an individual switch route for each interface. The end result is the same. All the switch routes defined in a single routing table will be connected together by CorePlus and no matter how interfaces are associated with the switch routes, transparency will exist between them.

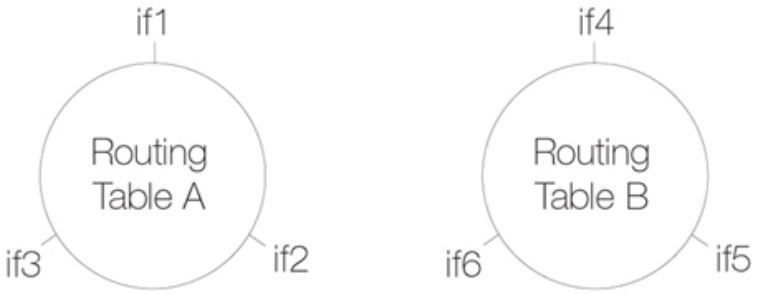
For example, if the interfaces *if1* to *if6* appear in a switch routes in routing table *A*, the resulting interconnections will be as illustrated below.



Connecting together switch routes in this way only applies, however, if all interfaces are associated with the same routing table. The situation where they are not, is described next.

Creating Separate Transparent Mode Networks

If we now have two routing tables *A* and *B* so that interfaces *if1*, *if2* and *if3* appear in a switch route in table *A* and interfaces *if4*, *if5*, *if6* appear in a switch route in table *B*, the resulting interconnections will be as illustrated below.



The diagram above illustrates how switch route interconnections for one routing table are completely separate from the switch route interconnections for another routing table. By using different routing tables in this way we can create two separate transparent mode networks.

The routing table used for an interface is decided by the *PBR Membership* parameter for each interface (*PBR* is short for *Policy Based Routing* which is the CorePlus term used for multiple routing tables). To implement separate Transparent Mode networks, interfaces must have their *PBR Membership* reset.

By default, all interfaces have *PBR membership* set to be *all* routing tables. By default, one *main* routing table always exists and once an additional routing table has been defined, the *PBR membership* for any interface can then be set to be that new table.

Transparent Mode with VLANs

If transparent mode is being set up for all hosts and users on a single VLAN then the technique described above of using multiple routing tables also applies. A dedicated routing table should be defined for a single VLAN and one switch route should then be defined in that routing table which refers to an interface group. The interface group needs to contain all the interfaces involved in the VLAN.

Enabling Transparent Mode Directly on Interfaces

The recommended way to enable Transparent Mode is to add switch routes, as described above. An alternative method is to enable transparent mode directly on an interface (a check box for this is provided in the graphical interfaces). When enabled in this way, default switch routes are automatically added to the routing table for the interface and any corresponding non-switch routes are automatically removed. This method is used in the detailed examples given later.

High Availability and Transparent Mode

Switch Routes cannot be used with High Availability and therefore true transparent mode cannot be implemented with a CorePlus High Availability Cluster.

Instead of Switch Routes the solution in a High Availability setup is to use Proxy ARP to separate two networks. This is described further in Section 5.2.5, “Proxy ARP”. The key disadvantage with this approach is that firstly, clients will not be able to roam between CorePlus interfaces, retaining the same IP address. Secondly, and more importantly, their network routes will need to be manually configured for proxy ARP.

Transparent Mode with DHCP

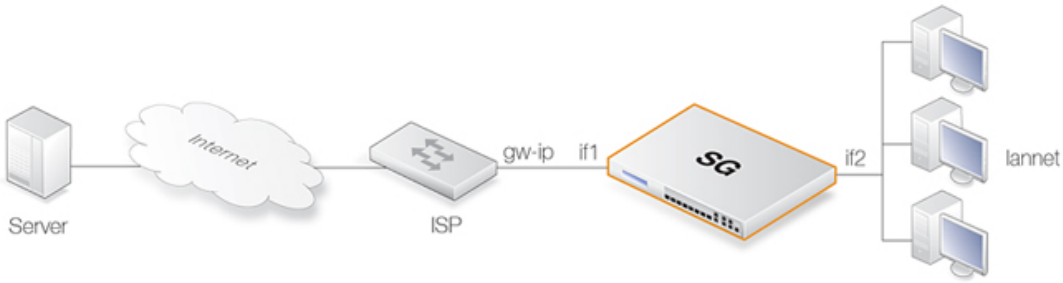
In most Transparent Mode scenarios, the IP address of users is predefined and fixed and is not dynamically fetched using DHCP. Indeed, the key advantage of Transparent Mode is that these users can plug in anywhere and CorePlus can route their traffic correctly after determining their whereabouts and IP address through ARP exchanges.

However, a DHCP server could be used to allocate user IP addresses in a Transparent Mode setup if desired. With Internet connections, it may be the ISP's own DHCP server which will hand out public IP addresses to users. In this case, CorePlus **MUST** be correctly configured as a *DHCP Relayer* to forward DHCP traffic between users and the DHCP server.

5.7.2. Enabling Internet Access

A common misunderstanding when setting up Transparent Mode is how to correctly set up access to the public Internet. Below is a typical scenario where a number of users on an IP network called *lannet* access the Internet via an ISP's gateway with IP address *gw-ip*.

Figure 5.8. Non-transparent Mode Internet Access

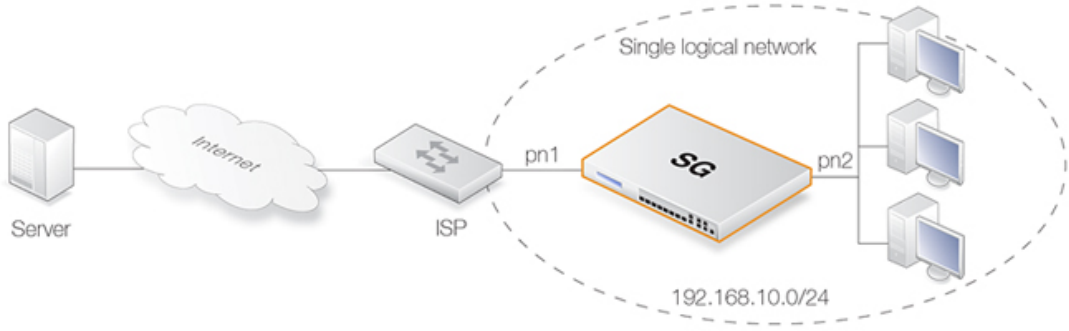


The non-switch route usually needed to allow Internet access would be:

Route type	Interface	Destination	Gateway
Non-switch	if1	all-nets	gw-ip

Now lets suppose the Clavister Security Gateway is to operate in transparent mode between the users and the ISP. The illustration below shows how, using switch routes, the Clavister Security Gateway is set up to be transparent between the internal physical Ethernet network (*pn2*) and the Ethernet network to the ISP's gateway (*pn1*). The two Ethernet networks are treated as a single logical IP network in Transparent Mode with a common address range (in this example *192.168.10.0/24*).

Figure 5.9. Transparent Mode Internet Access



In this situation, any "normal" non-switch *all-nets* routes in the routing table should be removed and replaced with an *all-nets* switch route (not doing this is a common mistake during setup). This switch route will allow traffic from the local users on Ethernet network *pn2* to find the ISP gateway.

These same users should also configure the internet gateway on their local computers to be the ISPs gateway address. In non-transparent mode the user's gateway IP would be the Clavister Security Gateway's IP address but in transparent mode the ISP's gateway is on the same logical IP network as the users and will therefore be *gw-ip*.

CorePlus May Also Need Internet Access

The Clavister Security Gateway also needs to find the public Internet if it is to perform CorePlus functions such as DNS lookup, Web Content Filtering or Anti-Virus and IDP updating. To allow this, individual "normal" non-switch routes need to be set up in the routing table for each IP address specifying the interface which leads to the ISP and the ISPs gateway IP address.

If the IP addresses that need to be reached by CorePlus are *85.12.184.39* and *194.142.215.15* then the complete routing table for the above example would be:

Route type	Interface	Destination	Gateway
Switch	if1	all-nets	
Switch	if2	all-nets	
Non-switch	if1	85.12.184.39	gw-ip
Non-switch	if1	194.142.215.15	gw-ip

The appropriate IP rules will also need to be added to the IP ruleset to allow Internet access through the Clavister Security Gateway.

Grouping IP Addresses

It can be quicker when dealing with many IP addresses to group all the addresses into a single group IP object and then use that object in a single defined route. In the above example, *85.12.184.39* and *194.142.215.15* could be grouped into a single object in this way.

Using NAT

NAT should not be enabled for CorePlus in Transparent Mode since, as explained previously, the Clavister Security Gateway is acting like a level 2 switch and address translation is done at the higher IP OSI layer.

The other consequence of not using NAT is that IP addresses of users accessing the Internet usually need to be public IP addresses.

If NATing needs to be performed in the example above to hide individual addresses from the

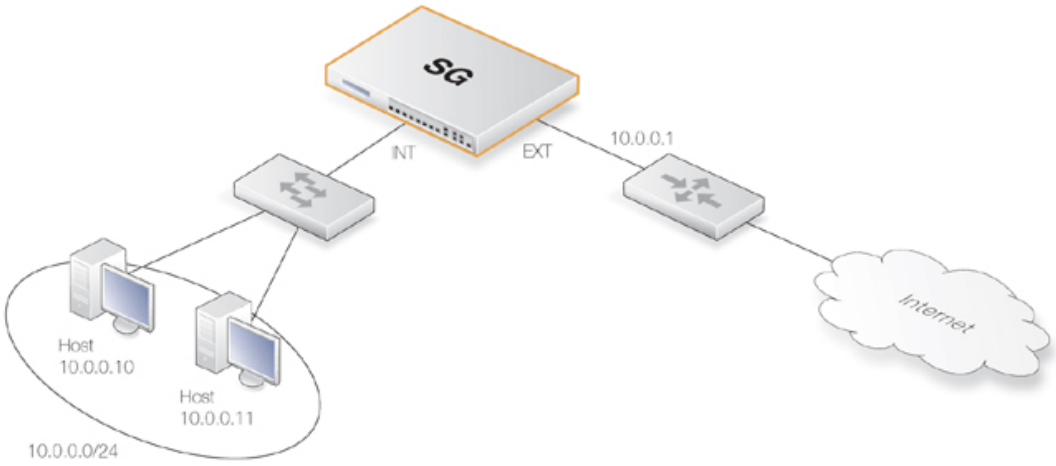
Internet, it would have to be done by a device (possibly another Clavister Security Gateway) between the *192.168.10.0/24* network and the public internet. In this case, internal IP addresses could be used by the users on Ethernet network *pn2*.

5.7.3. Transparent Mode Scenarios

Scenario 1

The security 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 security 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 5.10. Transparent Mode Scenario 1



Example 5.13. Setting up Transparent Mode for Scenario 1

Web Interface

Configure the interfaces:

1. Go to **Interfaces > Ethernet > Edit (wan)**
2. Now enter:
   - **IP Address:** 10.0.0.1
   - **Network:** 10.0.0.0/24
   - **Default Gateway:** 10.0.0.1
   - **Transparent Mode:** Enable
3. Click **OK**
4. Go to **Interfaces > Ethernet > Edit (lan)**
5. Now enter:
   - **IP Address:** 10.0.0.2
   - **Network:** 10.0.0.0/24
   - **Transparent Mode:** Enable

6. Click **OK**

Configure the rules:

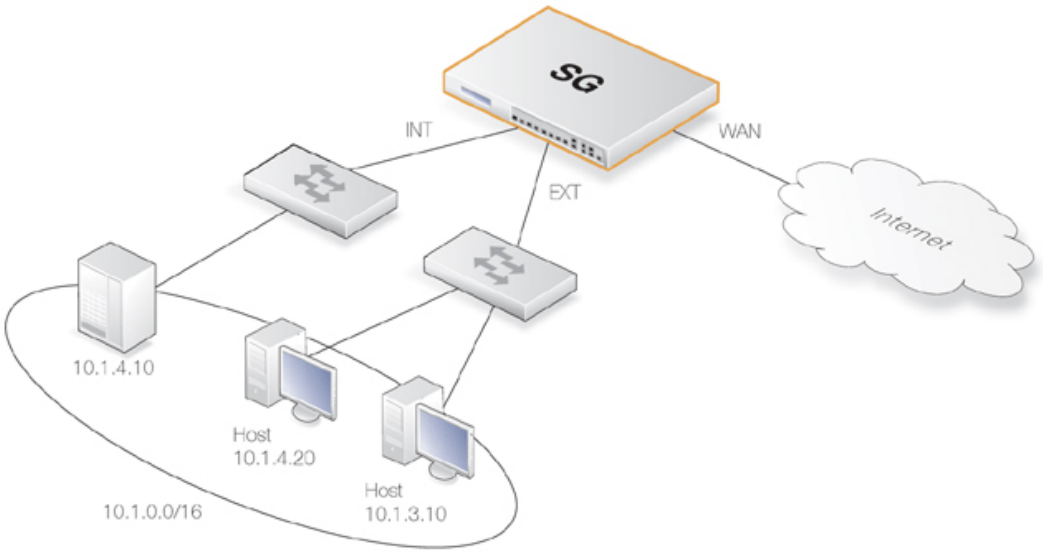
1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** HTTPAllow
   - **Action:** Allow
   - **Service:** http
   - **Source Interface:** lan
   - **Destination Interface:** any
   - **Source Network:** 10.0.0.0/24
   - **Destination Network:** all-nets (0.0.0.0/0)
3. Click **OK**

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 Clavister Security Gateway is transparent between the DMZ and LAN while traffic can subjected to the IP rule set.

Figure 5.11. Transparent Mode Scenario 2



Example 5.14. Setting up Transparent Mode for 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).

Web Interface

Configure the interfaces:

1. Go to **Interfaces > Ethernet > Edit (lan)**
2. Now enter:
   - **IP Address:** 10.0.0.1
   - **Network:** 10.0.0.0/24
   - **Transparent Mode:** Disable
   - **Add route for interface network:** Disable
3. Click **OK**
4. Go to **Interfaces > Ethernet > Edit (dmz)**
5. Now enter:
   - **IP Address:** 10.0.0.2
   - **Network:** 10.0.0.0/24
   - **Transparent Mode:** Disable
   - **Add route for interface network:** Disable
6. Click **OK**

Configure the interface groups:

1. Go to **Interfaces > Interface Groups > Add > InterfaceGroup**
2. Now enter:
   - **Name:** TransparentGroup
   - **Security/Transport Equivalent:** Disable
   - **Interfaces:** Select lan and dmz
3. Click **OK**

Configure the routing:

1. Go to **Routing > Main Routing Table > Add > SwitchRoute**
2. Now enter:
   - **Switched Interfaces:** TransparentGroup
   - **Network:** 10.0.0.0/24
   - **Metric:** 0
3. Click **OK**

Configure the rules:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** HTTP-LAN-to-DMZ

- **Action:** Allow
- **Service:** http
- **Source Interface:** lan
- **Destination Interface:** dmz
- **Source Network:** 10.0.0.0/24
- **Destination Network:** 10.1.4.10

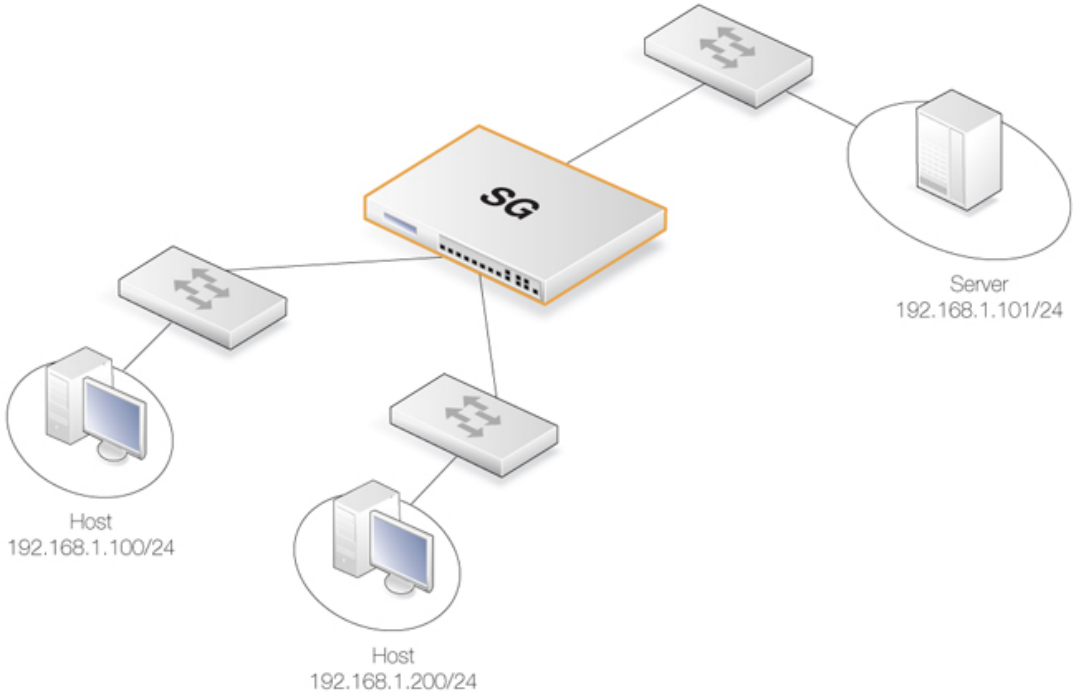
3. Click **OK**
4. Go to **Rules > IP Rules > Add > IPRule**
5. Now enter:
   - **Name:** HTTP-WAN-to-DMZ
   - **Action:** SAT
   - **Service:** http
   - **Source Interface:** wan
   - **Destination Interface:** dmz
   - **Source Network:** all-nets
   - **Destination Network:** ip_wan
   - **Translate:** Select Destination IP
   - **New IP Address:** 10.1.4.10
6. Click **OK**
7. Go to **Rules > IP Rules > Add > IPRule**
8. Now enter:
   - **Name:** HTTP-WAN-to-DMZ
   - **Action:** Allow
   - **Service:** http
   - **Source Interface:** wan
   - **Destination Interface:** dmz
   - **Source Network:** all-nets
   - **Destination Network:** ip_wan
9. Click **OK**

5.7.4. 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.

The diagram below 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 security gateway need to communicate and require CorePlus to relay switch BPDU messages in order that packets do not loop between the gateways.

Figure 5.12. An Example BPDU Relaying Scenario



Implementing BPDU Relaying

The CorePlus BDPU 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 BDPU messages to make sure the content type is supported. If it is not, the frame is dropped.

Enabling/Disabling BPDU Relaying

BPDU relaying is disabled by default and can be controlled through the advanced setting *Relay Spanning-tree BPDUs*. 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.

5.7.5. MPLS Pass Through

Multi-protocol Label Switching (MPLS) is a standard that allows the attaching of *labels* to IP packets to provide information about the packet's eventual destination. The router that initially attached the MPLS label is known as the *ingress router*.

Network nodes that support MPLS can then use this attached information to route packets without needing to perform route lookups and therefore increase processing speed. In addition to overall

faster traffic movement, MPLS also makes it easier to manage *Quality of Service* (QoS).

MPLS is considered to be "multi-protocol" because it works with the Internet Protocol, *Asynchronous Transport Mode* (ATM) and frame relay network. When considered in reference to the OSI network model, MPLS allows packets to be forwarded at the layer two level rather than at the layer three level and for this reason it is said to operate at the two and a half level.

CorePlus MPLS Support

CorePlus supports *MPLS Pass Through*. This is relevant in transparent mode scenarios where the MPLS labeled packets are allowed to traverse the Clavister Security Gateway. CorePlus can optionally validate the integrity of these MPLS packets and the administrator can change the advanced setting *Relay MPLS* to specify the specific action to be taken. The possible values for this setting are:

- **Ignore** - Verify packets and allow all verified MPLS labeled packets to pass silently. Packets that fail verification are logged.
- **Log** - Verify packets and allow all verified MPLS packets to pass as well as being logged. Packets that fail verification are also logged.
- **Drop** - Silently drop all MPLS packets without verification or logging.
- **Drop/Log** - Drop all MPLS packets without verification and log these drops.

5.7.6. Advanced Settings for Transparent Mode

CAM To L3 Cache Dest Learning

Enable this if the security gateway should be able to learn the destination for hosts by combining destination address information and information found in the CAM table.

Default: *Enabled*

Decrement TTL

Enable this if the TTL should be decremented each time a packet traverses the security gateway in Transparent Mode.

Default: *Disabled*

Dynamic CAM Size

This setting can be used to manually configure the size of the CAM table. Normally *Dynamic* is the preferred value to use.

Default: *Dynamic*

CAM Size

If the Dynamic CAM Size setting is not enabled then this is the maximum number of entries in each CAM table.

Default: *8192*

Dynamic L3C Size

Allocate the L3 Cache Size value dynamically.

Default: *Enabled*

L3 Cache Size

This setting is used to manually configure the size of the Layer 3 Cache. Enabling *Dynamic L3C Size* is normally preferred.

Default: *Dynamic*

Transparency ATS Expire

Defines the lifetime of an unanswered ARP Transaction State (ATS) entry in seconds. Valid values are 1-60 seconds.

Default: *3 seconds*

Transparency ATS Size

Defines the maximum total number of ARP Transaction State (ATS) entries. Valid values are 128-65536 entries.

Default: *4096*

Note

*Both **Transparency ATS Expire** and **Transparency ATS Size** can be used to adjust the ATS handling to be optimal in different environments.*

Null Enet Sender

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*

Broadcast Enet Sender

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*

Multicast Enet Sender

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*

Relay Spanning-tree BPDUs

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* - Let the packets pass but do not log
- *Log* - Let the packets pass and log the event
- *Drop* - Drop the packets
- *DropLog* - Drop packets log the event

Default: *Drop*

Relay MPLS

When set to **Ignore** all incoming MPLS packets are relayed in transparent mode. Options:

- *Ignore* - Let the packets pass but do not log
- *Log* - Let the packets pass and log the event
- *Drop* - Drop the packets
- *DropLog* - Drop packets log the event

Default: *Drop*

Chapter 6. DHCP Services

This chapter describes DHCP services in CorePlus.

6.1. Overview

Dynamic Host Configuration Protocol (DHCP) is a protocol that allows network administrators to automatically assign IP numbers to computers on a network.

IP Address Assignment

A *DHCP Server* implements the task of assigning IP addresses to DHCP clients. These addresses come from a pre-defined IP address pool which DHCP manages. 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.

DHCP Leases

Compared to static assignment, where the client owns the address, dynamic addressing by a DHCP server leases the address to each client for a pre-defined period of time. During the lifetime of a 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 the assigned 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 and release the IP address.

The lease time can be configured in a DHCP server by the administrator.

6.2. DHCP Servers

DHCP servers assign and manage the IP addresses taken from a specified address pool. In CorePlus, DHCP servers are not limited to serving a single range of IP addresses but can use any IP address range that can be specified by a CorePlus IP address object.

The administrator has the ability to set up one or more logical DHCP servers in CorePlus. Filtering of DHCP client requests to different DHCP servers is based on a combination of:

- **Interface** - Each CorePlus interface can have, at most, one single logical DHCP server associated with it. In other words, CorePlus can provision DHCP clients using different address ranges depending on what interface they are located on.
- **Relayer IP** - The relayer IP address in the IP packet is also used to determine the server. The default value of *all-nets* means that this all addresses are accepted and only the interface is considered in making a DHCP server selection. The other options for this parameter are described further below.

Multiple DHCP servers form a list as they are defined, the last defined being at the top of the list. When CorePlus searches for a DHCP server to service a request, it goes through the list from top to bottom and chooses the first server with a matching combination of interface and relayer IP filter value. If there is no match in the list then the request is ignored.

Using Relayer IP Address Filtering

As explained above a DHCP server is selected based on a match of both interface and relayer IP filter. Each DNS server must have a relayer IP filter value specified and the possible values are as follows:

- The default value is *all-nets* (*0.0.0.0/0*). This means all DHCP requests will match this filter value regardless if the DHCP requests comes from a client on the local network or has arrived via a DHCP relayer.
- A value of *0.0.0.0* will match DHCP requests that come from a local client only. DHCP requests that have been relayed by a DHCP relayer will be ignored.
- A specific IP address. This is the IP address of the DHCP relayer through which the DHCP request has come. Requests from local clients or other DHCP relayers will be ignored.

Configurable DHCP Options

A number of options which relate to the response sent to clients can be configured for each DHCP server instance:

- **Netmask** - netmask sent to the DHCP Client.
- **Gateway Address** - what IP should be sent to the client for use as the default gateway. If 0.0.0.0 is specified the IP given to the client will be sent as the gateway.
- **Domain Name** - The domain which the client will belong to.
- **Lease Time** - the time, in seconds that a DHCP lease should be provided to a host after which the client must renew the lease.
- **DNS Servers** - DNS servers the client can use for DNS lookup.
- **WINS Servers** - WINS servers the client can use for WINS lookup.

- **Next Server** - the IP address of the next server in the boot process, this is usually a TFTP server.

In addition, *Custom Options* can be specified in order to have the DHCP servers hand out all options supported by the DHCP standard.

Example 6.1. Setting up a DHCP server

This example shows how to set up a DHCP server called *DHCPServer1* which assigns and manages IP addresses from an IP address pool called *DHCPRange1*. This example assumes you have created an IP range for the DHCP Server.

CLI

```
Device:/> add DHCPServer DHCPServer1 Interface=lan
     IPAddressPool=DHCPRange1 Netmask=255.255.255.0
```

Web Interface

1. Go to **System > DHCP > DHCP Servers >Add > DHCPServer**
2. Now enter:
   - **Name:** DHCPServer1
   - **Interface Filter:** lan
   - **IP Address Pool:** DHCPRange1
   - **Netmask:** 255.255.255.0
3. Click **OK**

Example 6.2. Checking the status of a DHCP server

CLI

To see the status of all servers:

```
Device:/> dhcpserver
```

To list all configured servers:

```
Device:/> show dhcpserver
```

Web Interface

Go to **Status > DHCP Server** in the menu bar - a list of all servers will be displayed with the status

Tip
DHCP leases are remembered by CorePlus between system restarts.

6.3. Static DHCP Assignment

Where the administrator requires a fixed relationship between a client and the assigned IP address, CorePlus allows the assignment of a given IP to a specific MAC address.

Example 6.3. Setting up Static DHCP

This example shows how to assign the IP address *192.168.1.1* to the MAC address *00-90-12-13-14-15*. The examples assumes that the DHCP server *DHCPServer1* has already been defined.

CLI

1. First, change the category to the *DHCPServer1* context:

```
Device:/> cc DHCPServer DHCPServer1
```

2. Add the static DHCP assignment:

```
Device:/> add DHCPServerPoolStaticHost Host=192.168.1.1
                                                MACAddress=00-90-12-13-14-15
```

3. All static assignments can then be listed and each is listed with an index number:

```
Device:/> show

   #  Comments
   -  -------
+  1  (none)
```

4. An individual static assignment can be shown using its index number:

```
Device:/> show DHCPServerPoolStaticHost 1

   Property  Value
-----------  -----------------
     Index:  1
      Host:  192.168.1.1
MACAddress:  00-90-12-13-14-15
  Comments:  (none)
```

5. The assignment could be changed later to IP address *192.168.1.12* with the following command:

```
Device:/> set DHCPServerPoolStaticHost 1 Host=192.168.1.12
                                                MACAddress=00-90-12-13-14-15
```

Web Interface

1. Go to **System > DHCP > DHCP Servers > DHCPServer1 > Static Hosts > Add > Static Host Entry**
2. Now enter:
   - **Host:** 19.168.1.1
   - **MAC:** 00-90-12-13-14-15
3. Click **OK**

6.3.1. DHCP Advanced Settings

The following advanced settings are available with DHCP servers.

Auto Save Policy

What policy should be used to save the lease database to the disk, possible settings are *Disabled*, *ReconfShut* or *ReconfShutTimer*.

Default: *ReconfShut*

Lease Store Interval

How often, in seconds, the leases database should be saved to disk if *DHCPServer_SaveLeasePolicy* is set to *ReconfShutTimer*.

Default: *86400*

6.4. DHCP Relaying

The Problem

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 always need to be on the same physical network. In a large Internet-like network topology, this means there would have to be a different DHCP server on every network. This problem is solved by the use of a DHCP relayer.

The Relayer Solution

A DHCP relayer takes the place of the DHCP server in the local network and acts as the link between the client and a remote DHCP server. It intercepts requests from clients and relays them to the server. The server then responds to the relayer, which forwards the response back to the client. DHCP relayers use the TCP/IP *Bootstrap Protocol* (BOOTP) to implement relay functionality. For this reason they are sometimes referred to as *BOOTP relay agents*.

Example 6.4. Setting up a DHCP Relayer

This example allows clients on VLAN interfaces to obtain IP addresses from a DHCP server. It is assumed the security 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.

CLI

1. Add the VLAN interfaces *vlan1* and *vlan2* that should relay to an interface group called *ipgrp-dhcp*:

```
Device:/> add Interface InterfaceGroup ipgrp-dhcp Members=vlan1,vlan2
```

2. Add a DHCP relayer called *vlan-to-dhcpserver*:

```
Device:/> add DHCPRelay vlan-to-dhcpserver Action=Relay TargetDHCPServer=ip-dhcp
     SourceInterface=ipgrp-dhcp AddRoute=Yes ProxyARPInterfaces=ipgrp-dhcp
```

Web Interface

Adding VLAN interfaces *vlan1* and *vlan2* that should relay to an interface group named as *ipgrp-dhcp*:

1. Go to **Interface > Interface Groups > Add > InterfaceGroup**
2. Now enter:
   - **Name:** ipgrp-dhcp
   - **Interfaces**: select *vlan1* and *vlan2* from the **Available** list and put them into the **Selected** list.
3. Click **OK**

Adding a DHCP relayer called as *vlan-to-dhcpserver*:

1. Go to **System > DHCP > Add > DHCP Relay**
2. Now enter:
   - **Name:** vlan-to-dhcpserver
   - **Action:** Relay
   - **Source Interface:** ipgrp-dhcp
   - **DHCP Server to relay to:** ip-dhcp

- **Allowed IP offers from server:** all-nets

3. Under the **Add Route** tab, check **Add dynamic routes for this relayed DHCP lease**
4. Click **OK**

6.4.1. DHCP Relay Advanced Settings

The following advanced settings are available with DHCP relaying.

Max Transactions

Maximum number of transactions at the same time.

Default: *32*

Transaction Timeout

For how long a dhcp transaction can take place.

Default: *10 seconds*

Max PPM

How many dhcp-packets a client can send to through CorePlus to the dhcp-server during one minute.

Default: *500 packets*

Max Hops

How many hops the dhcp-request can take between the client and the dhcp-server.

Default: *5*

Max lease Time

The maximum leastime allowed through CorePlus, if the DHCP server has a higher lease time, it will be reduced down to this value.

Default: *10000 seconds*

Max Auto Routes

How many relays that can be active at the same time.

Default: *256*

Auto Save Policy

What policy should be used to save the relay list to the disk, possible settings are *Disabled*, *ReconfShut*, or *ReconfShutTimer*.

Default: *ReconfShut*

Auto Save Interval

How often, in seconds, should the relay list be saved to disk if *DHCPServer_SaveRelayPolicy* is set to ReconfShutTimer.

Default: *86400*

6.5. IP Pools

Overview

IP pools are used to offer other subsystems access to a cache of DHCP IP addresses. These addresses are gathered into a pool by internally maintaining a series of DHCP clients (one per IP). The DHCP servers used by a pool can either be external or be DHCP servers defined in CorePlus itself. External DHCP servers can be specified as the server on a specific interface or by a unique IP address. Multiple IP Pools can be set up with different identifying names.

The primary usage of IP Pools is with *IKE Config Mode* which a feature used for allocating IP addresses to remote clients connecting through IPsec tunnels. For more information on this see Section 10.4.3.4, “Using Config Mode”.

Basic IP Pool Options

The basic options available for an IP Pool are:

DHCP Server behind interface — Indicates that the IP pool should use the DHCP server(s) residing on the specified interface.

Server filter — Optional setting used to specify which servers to use. If unspecified any DHCP server on the interface will be used. The order of the provided address or ranges (if multiple) will be used to indicate the preferred servers.

Specify DHCP Server Address — Specify DHCP server IP(s) in preferred ascending order to be used. Using the IP loopback address *127.0.0.1* indicates that the DHCP server is CorePlus itself.

Client IP filter — Optional setting used to specify which offered IPs are valid to use. In most cases this will be set to the default of **all-nets**. Alternatively a set of IP ranges might be specified. The filter ensures that only certain IP addresses from DHCP servers are acceptable and is used in the situation where there might be a DHCP server response with an unacceptable IP address.

Advanced IP Pool Options

Advanced options available for IP Pool configuration are:

Routing table — Policy routing table to be used for lookups when resolving the destination interfaces for the configured DHCP servers.

Receive interface — "Simulated" receive interface. This can be used in policy based routing rules and/or used to trigger a specific DHCP server rule if the pool is using a DHCP server in CorePlus and the IP address of that server has been specified as the loopback interface.

MAC Range — A range of MAC addresses that will be use to create "fake" DHCP clients. Used when the DHCP server(s) map clients by the MAC address. An indication of the need for MAC ranges is when the DHCP server keeps giving out the same IP for each client.

Prefetched leases — Specifies the number of leases to keep prefetched. Prefetching will improve performance since there will not be any wait time when a system requests an IP (while there exists prefetched IPs).

Maximum free The maximum number of "free" IPs to be kept. Must be equal to or greater than the prefetch parameter. The pool will start releasing (giving back IPs to the DHCP server) when the number of free clients exceeds this value.

Maximum clients Optional setting used to specify the maximum number of clients (IPs) allowed in the pool.

Using Prefetched Leases

As mentioned in the previous section, the *Prefetched Leases* option specifies the size of the cache of leases which is maintained by CorePlus. This cache provides fast lease allocation and can improve overall system performance. It should be noted however that the entire prefetched number of leases is requested at system startup and if this number is too large then this can degrade initial performance.

As leases in the prefetch cache are allocated, requests are made to DHCP servers so that the cache is always full. The administrator therefore has to make a judgement as to the optimal initial size of the prefetch cache.

Example 6.5. Creating an IP Pool

This example shows the creation of an IP Pool object that will use the DHCP server on IP address *28.10.14.1* with 10 prefetched leases. It is assumed that this IP address is already defined in the address book as an IP object called *ippool_dhcp*

CLI

```
Device:/> add IPPool ip_pool_1 DHCPServerType=ServerIP ServerIP=ippool_dhcp
```

Web Interface

1. Go to **Objects > IP Pools > Add > IP Pool**
2. Now enter **Name:** ip_pool_1
3. Select **Specify DHCP Server Address**
4. Add *ippool_dhcp* to the **Selected** list
5. Select the **Advanced** tab
6. Set **Prefetched Leases** to *10*
7. Click **OK**

Chapter 7. Security Mechanisms

This chapter describes CorePlus security features.

7.1. Access Rules

7.1.1. 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 checks the connection source against a set of *Access Rules*. Access Rules can 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.

The Default Access Rule

Even if the administrator does not explicitly specify any Access Rules, a basic access rule is always in place which is known as the *Default Access Rule*. This default rule always checks incoming traffic by performing a reverse lookup in the routing tables. This lookup validates that the incoming traffic is coming from a source that the routing tables indicate is accessible via the interface on which the traffic arrived. If this reverse lookup fails then the connection is dropped and a "Default Access Rule" log message will be generated.

For most configurations the Default Access Rule is sufficient and the administrator does not need to explicitly specify other rules. The default rule can, for instance, protect against IP spoofing, which is described in the next section. If Access Rules are explicitly specified, then the Default Access Rule is still applied if a new connection does not match any of the specified rules.

7.1.2. 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 because of its nature.

VPNs provide one means of avoiding spoofing but where a VPN is not an 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.

7.1.3. 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 Actions

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.

Turning Off *Default Access Rule* Messages

If, for some reason, the "Default Access Rule" log message is continuously being generated by some source and needs to be turned off, then the way to do this is to specify an Access Rule for that source with an action of **Drop**.

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 7.1. 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.

CLI

```
Device:/> add Access Name=lan_Access Interface=lan Network=lannet Action=Expect
```

Web Interface

1. Go to **Rules > Access**
2. Select **Access Rule** in the **Add menu**
3. Now enter:
   - **Name**: lan_Access
   - **Action**: Except
   - **Interface**: lan
   - **Network**: lannet
4. Click **OK**

7.2. ALGs

7.2.1. 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 object acts as a mediator in accessing commonly used Internet applications outside the protected network, for example 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 protocol and perform checks at the higher levels of the TCP/IP stack.

ALGs exist for the following protocols in CorePlus:

- HTTP
- FTP
- TFTP
- SMTP
- POP3
- SIP
- H.323

Deploying an ALG

Once a new ALG object is defined by the administrator, it is brought into use by first associating it with a *Service* object and then associating that Service with an IP rule in the CorePlus IP rule set.

Figure 7.1. Deploying an ALG Object

Maximum Connection Sessions

The Service associated with an ALG has a configurable parameter associated with it called *Max Sessions* and the default value varies according to the type of ALG. For instance, the default value for the HTTP ALG is *1000*. This means that a 1000 connections are allowed in total for the HTTP Service across all interfaces. The full list of default maximum session values are:

- HTTP ALG - *1000* sessions.
- FTP ALG - *200* sessions.
- TFTP ALG - *200* sessions.
- SMTP ALG - *200* sessions.
- POP3 ALG - *200* sessions.
- H.323 ALG - *100* sessions.
- SIP ALG - *200* sessions.

Note
This default value can often be too low for HTTP if there are large number of clients connecting through the Clavister Security Gateway and it is therefore recommended to consider using a higher value in such circumstances.

ALGs and Syn Flood Protection

It should be noted that user-defined custom Service objects have the option to enable *Syn Flood Protection*, a feature which specifically targets Syn Flood attacks. If this option is enabled for a Service object then any ALG associated with that Service will not be used.

7.2.2. The HTTP ALG

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. A client, such as a Web browser, sends a request by establishing a TCP/IP connection to a known 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 perhaps an error message.

The HTTP protocol has particular issues associated with it because of the wide variety of web sites that exist and because of the range of file types that can be downloaded using the protocol.

HTTP ALG Features

The HTTP ALG is an extensive CorePlus subsystem consisting of the options described below:

- **Static Content Filtering** - This deals with *Blacklisting* and *Whitelisting* of specific URLs.
  - ***URL Blacklisting***
    Specific URLs can be blacklisted so that they are not accessible. Wildcarding can be used when specifying URLs, as described below.

- ***URL Whitelisting***
  The opposite to blacklisting, this makes sure certain URLs are always allowed. Wildcarding can also be used for these URLs, as described below.

  It is important to note that whitelisting a URL means that it cannot be blacklisted and it also cannot be dropped by web content filtering (if that is enabled, although it will be logged). Anti-Virus scanning, if it is enabled, is always applied to the HTTP traffic even if it is whitelisted.

These features are described in depth in Section 7.3.3, “Static Content Filtering”.

- **Dynamic Content Filtering** - Access to specific URLs can be allowed or blocked according to policies for certain types of web content. Access to news sites might be allowed whereas access to gaming sites might be blocked.

  This feature is described in depth in Section 7.3.4, “Dynamic Web Content Filtering”.

- **Anti-Virus Scanning** - The contents of HTTP file downloads can be scanned for viruses. Suspect files can be dropped or just logged.

  This feature is common to a number of ALGs and is described fully in Section 7.4, “Anti-Virus Scanning”.

- **Verify File Integrity** - This part of the ALG deals with checking the filetype of downloaded files. There are two separate optional features with filetype verification: **Verify MIME type** and **Allow/Block Selected Types**, and these are described below:

  1. **Verify MIME type**

     This option enables checking that the filetype of a file download agrees with the contents of the file (the term *filetype* here is also known as the *filename extension*).

     All filetypes that are checked in this way by CorePlus are listed in Appendix C, *Verified MIME filetypes*. When enabled, any file download that fails MIME verification, in other words its filetype does not match its contents, is dropped by CorePlus on the assumption that it can be a security threat.

  2. **Allow/Block Selected Types**

     This option operates independently of the MIME verification option described above but is based on the predefined filetypes listed in Appendix C, *Verified MIME filetypes*. When enabled, the feature operates in either a *Block Selected* or an *Allow Selected* mode. These two modes function as follows:

     i. Block Selected

     The filetypes marked in the list will be dropped as downloads. To make sure that this is not circumvented by renaming a file, CorePlus looks at the file's contents (in a way similar to MIME checking) to confirm the file is what it claims to be.

     If, for example, *.exe* files are blocked and a file with a filetype of *.jpg* (which is not blocked) is found to contain *.exe* data then it will be blocked. If blocking is selected but nothing in the list is marked, no blocking is done.

     ii. Allow Selected

     Only those filetypes marked will be allowed in downloads and other will be dropped. As with blocking, file contents are also examined to verify the file's contents. If, for example, *.jpg* files are allowed and a file with a filetype of *.jpg* is found to contain *.exe* data then the download will be dropped. If nothing is marked in this mode then no files can be downloaded.

     Additional filetypes not included by default can be added to the Allow/Block list however

these cannot be subject to content checking meaning that the file extension will be trusted as being correct for the contents of the file.

Note

*The **Verify MIME type** and **Allow/Block Selected Types** options work in the same way for the FTP, POP3 and SMTP ALGs.*

- **Download File Size Limit** - A file size limit can additionally be specified for any single download (this option is available only for HTTP and SMTP ALG downloads).

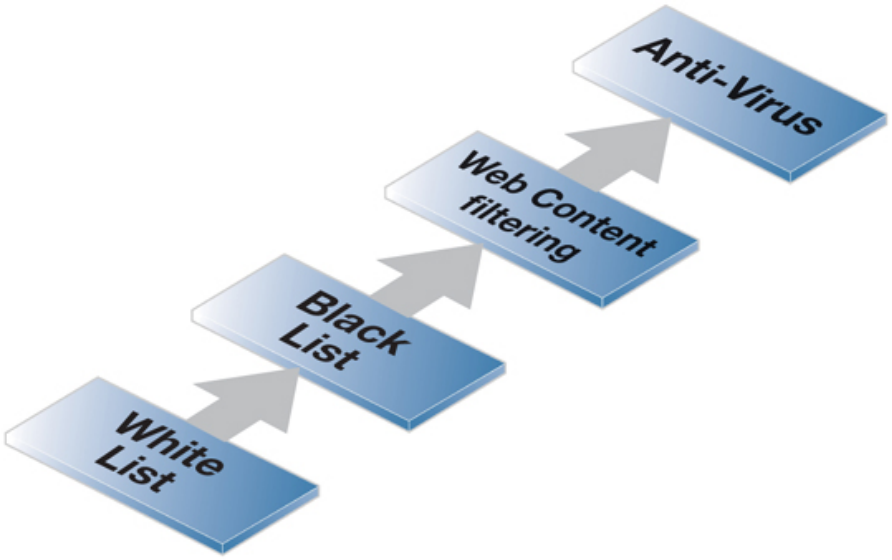
The Ordering for HTTP Filtering

HTTP filtering obeys the following processing order and is similar to the order followed by the SMTP ALG:

1. Whitelist.
2. Blacklist.
3. Web content filtering (if enabled).
4. Anti-virus scanning (if enabled).

As described above, if a URL is found on the whitelist then it will not be blocked if it also found on the blacklist. If it is enabled, Anti-virus scanning is always applied, even though a URL is whitelisted. If it is enabled, Web content filtering is still applied to whitelisted URLs but if instead of blocking, flagged URLs are only logged. If it is enabled, Anti-virus scanning is always applied, even though a URL is whitelisted.

Figure 7.2. HTTP ALG Processing Order



Using Wildcards in White and Blacklists

Entries made in the white and blacklists can make use of *wildcarding* to have a single entry be

equivalent to a large number of possible URLs. The wildcard character "*" can be used to represent any sequence of characters.

For example, the entry **.some_domain.com* will block all pages whose URLs end with *some_domain.com*.

If we want to now explicitly allow one particular page then this can be done with an entry in the whitelist of the form *my_page.my_company.com* and the blacklist will not prevent this page from being reachable since the whitelist has precedence.

Deploying an HTTP ALG

As mentioned in the introduction, the HTTP ALG object is brought into use by first associating it with a Service object and then associating that Service object with an IP rule in the IP rule set. A number of pre-defined HTTP Services could be used with the ALG. For example, the **http** service might be selected for this purpose. As long as the associated Service is associated with an IP rule then the ALG will be applied to traffic targeted by that IP rule.

The **https** Service (which is also included in the **http-all** Service) cannot be used with an HTTP ALG since HTTPS traffic is encrypted.

7.2.3. The FTP ALG

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 FTP mode being used.

Connection Modes

FTP operates in two modes: *active* and *passive*. These determine the role of the server when opening data channels between client and server.

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 *active* and *passive* modes of FTP operation present problems for Clavister Security Gateways. Consider a scenario where an FTP client on the internal network connects through the security 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 doesn't know 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 security 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 security 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.

Filetype Checking

The FTP ALG offers the same filetype verification for downloaded files that is found in the HTTP ALG. This consists of two separate options:

- **MIME Type Verification**
  When enabled, CorePlus checks that a download's stated filetype matches the file's contents. Mismatches result in the download being dropped.
- **Allow/Block Selected Types**
  If selected in blocking mode, specified filetypes are dropped when downloaded. If selected in allow mode, only the specified filetypes are allowed as downloads. CorePlus also performs a check to make sure the filetype matches the contents of the file. New filetypes can be added to the predefined list of types.

The above two options for filetype checking are the same as those available in the HTTP ALG and are more fully described in Section 7.2.2, “The HTTP ALG”.

Anti-Virus Scanning

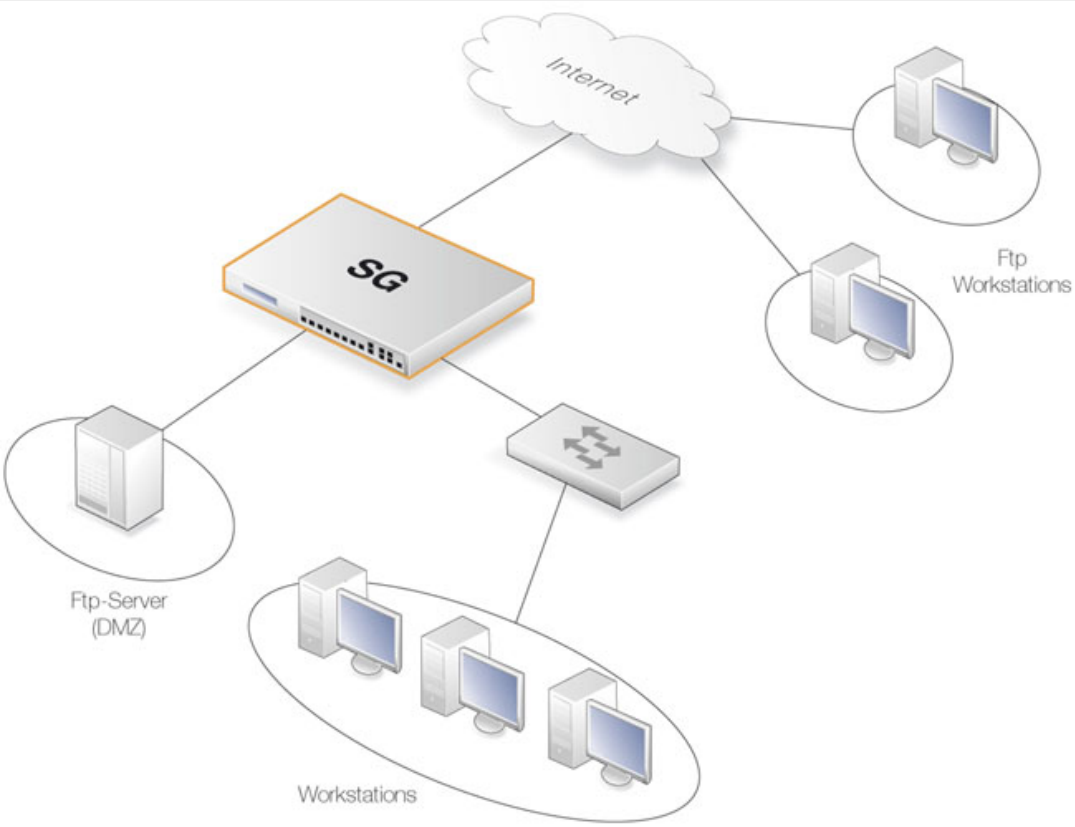
The CorePlus Anti-Virus subsystem can be enabled to scan all FTP downloads searching for malicious code. Suspect files can be de dropped or just logged.

This feature is common to a number of ALGs and is described fully in Section 7.4, “Anti-Virus

Scanning".

Example 7.2. Protecting an FTP Server with an 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:

Web Interface

A. Define the ALG:

1. Go to **Objects > ALG > Add > FTP ALG**
2. Enter **Name:** ftp-inbound
3. Check **Allow client to use active mode**
4. Uncheck **Allow server to use passive mode**
5. Click **OK**

B. Define the Service:

1. Go to **Objects > Services > Add > TCP/UDP Service**
2. Enter the following:
   - **Name:** ftp-inbound
   - **Type:** select TCP from the list
   - **Destination:** 21 (the port the FTP server resides on)
   - **ALG:** select the "ftp-inbound" that has been created

3. Click **OK**

C. Define a rule to allow connections to the public IP on port 21 and forward that to the internal FTP server:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** SAT-ftp-inbound
   - **Action:** SAT
   - **Service:** ftp-inbound
3. For **Address Filter** enter:
   - **Source Interface:** any
   - **Destination Interface:** core
   - **Source Network:** all-nets
   - **Destination Network:** ip_wan (assuming the external interface has been defined as this)
4. For **SAT** check **Translate the Destination IP Address**
5. Enter **To: New IP Address:** ftp-internal (assume this internal IP address for FTP server has been defined in the Address Book object)
6. **New Port:** 21
7. Click **OK**

D. Traffic from the internal interface needs to be NATed:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** NAT-ftp
   - **Action:** NAT
   - **Service:** ftp-inbound
3. For **Address Filter** enter:
   - **Source Interface:** dmz
   - **Destination Interface:** core
   - **Source Network:** dmznet
   - **Destination Network:** ip_wan
4. For **NAT** check **Use Interface Address**
5. Click **OK**

E. Allow incoming connections (SAT requires a second Allow rule):

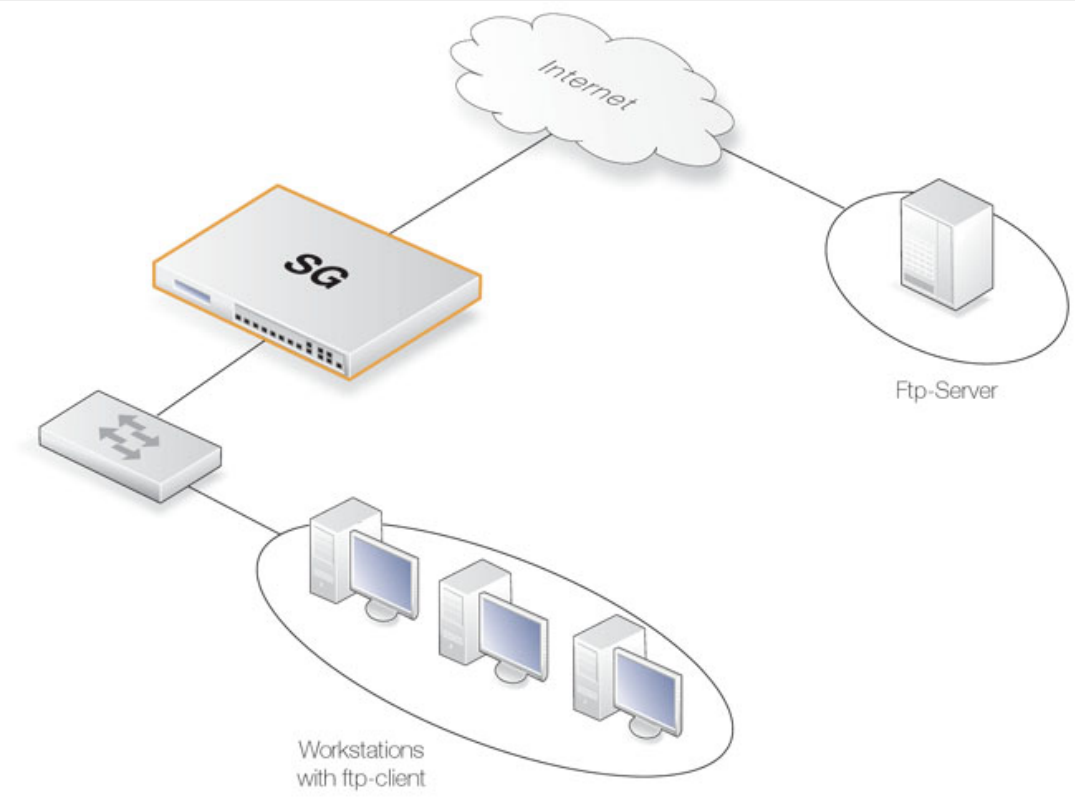
1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** Allow-ftp
   - **Action:** Allow
   - **Service:** ftp-inbound
3. For **Address Filter** enter:
   - **Source Interface:** any
   - **Destination Interface:** core

- **Source Network:** all-nets
- **Destination Network:** ip_wan

4. Click **OK**

Example 7.3. 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:

Web Interface

A. Create the FTP ALG:

1. Go to **Objects > ALG > Add > FTP ALG**
2. Enter **Name:** ftp-outbound
3. Uncheck **Allow client to use active mode**
4. Check **Allow server to use passive mode**
5. Click **OK**

B. Create the Service:

1. Go to **Objects > Services > Add > TCP/UDP Service**
2. Now enter:
   - **Name:** ftp-outbound

- **Type:** select TCP from the dropdown list
- **Destination:** 21 (the port the ftp server resides on)
- **ALG:** select the newly created *ftp-outbound*

3. 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.

C. Allow connections to ftp-servers on the outside:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** Allow-ftp-outbound
   - **Action:** Allow
   - **Service:** ftp-outbound
3. For **Address Filter** enter:
   - **Source Interface:** lan
   - **Destination Interface:** wan
   - **Source Network:** lannet
   - **Destination Network:** all-nets
4. Click **OK**

D. Rules (Using Private IPs). If the security gateway is using private IP's, the following NAT rule need to be added instead:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** NAT-ftp-outbound
   - **Action:** NAT
   - **Service:** ftp-outbound
3. For **Address Filter** enter:
   - **Source Interface:** lan
   - **Destination Interface:** wan
   - **Source Network:** lannet
   - **Destination Network:** all-nets
4. Check **Use Interface Address**
5. Click **OK**

7.2.4. The TFTP ALG

Trivial File Transfer Protocol (TFTP) is a much simpler version of FTP with more limited capabilities. Its purpose is to allow a client to upload files to or download files from a host system. TFTP data transport is based on the UDP protocol and therefore it supplies its own transport and session control protocols which are layered onto UDP.

TFTP is widely used in enterprise environments for updating software and backing up configurations on network devices. TFTP is recognized as being an inherently insecure protocol and its usage is often confined to internal networks. The CorePlus ALG provides an extra layer of security to TFTP in being able to put restrictions on its use.

General TFTP Options

Allow/Disallow Read — The TFTP **GET** function can be disabled so that files cannot be retrieved by a TFTP client. The default value is *Allow*.

Allow/Disallow Write — The TFTP **PUT** function can be disabled so that files cannot be written by a TFTP client. The default value is *Allow*.

Remove Request Option — Specifies if options should be removed from request. The default is *False* which means "do not remove".

Block Unknown Options — This option allows the blocking of any option in a request other than the blocksize, the timeout period and the file transfer size. The default is *False* which means "do not block".

TFTP Request Options

As long as the **Remove Request Option** described above is set to *false* (options are not removed) then the following request option settings can be applied:

Maximum Blocksize — The maximum blocksize allowed can be specified. The allowed range is 0 to 65464 bytes. The default value is 65464 bytes.

Maximum File Size — The maximum size of a file transfer can be restricted. By default this is the absolute maximum allowed which 999,999 Kbytes.

Allow Directory Traversal — This option can disallow directory traversal through the use of filenames containing consecutive periods ("..").

Allowing Request Timeouts

The CorePlus TFTP ALG blocks the repetition of an TFTP request coming from the same source IP address and port within a fixed period of time. The reason for this is that some TFTP clients might issue requests from the same source port without allowing an appropriate timeout period.

7.2.5. The SMTP ALG

Simple Mail Transfer Protocol (SMTP) is a text based protocol used for transferring email between mail servers over the Internet. Typically the local SMTP server will be located on a DMZ so that mail sent by remote SMTP servers will traverse the Clavister Security Gateway to reach the local server (this setup is illustrated later in Section 7.2.5.1, “DNSBL SPAM Filtering”). Local users will then use email client software to retrieve their email from the local SMTP server.

SMTP is also used when clients are sending email and the SMTP ALG can be used to monitor SMTP traffic originating from both clients and servers.

SMTP ALG Options

Key features of the SMTP ALG are:

Email Rate Limiting

A maximum allowable rate of email messages can be specified. This rate is calculated on a *per source IP address* basis, in other words it is not the total rate that is of interest but the rate from a certain email source.

This is a very useful feature to have since it is possible to put in a block against either an infected client or an infected server sending large amounts of malware generated emails.

Email Size Limiting

A maximum allowable size of email messages can be specified. This feature counts the total amount of bytes sent for a single email which is the header size plus body size plus the size of any email attachments after they are encoded. It should be kept in mind that an email with, for example, an attachment of 100 Kbytes, will be larger than 100 Kbytes. The transferred size might be 120 Kbytes or more since the encoding which takes place automatically for attachments may substantially increase the transferred attachment size.

The administrator should therefore add a reasonable margin above the anticipated email size when setting this limit.

Email address blacklisting

A blacklist of sender or recipient email addresses can be specified so that mail from/to those addresses is blocked. The blacklist is applied after the whitelist so that if an address matches a whitelist entry it is not then checked against the blacklist.

Email address whitelisting

A whitelist of email addresses can be specified so that any mail from/to those addresses is allowed to pass through the ALG regardless if the address is on the blacklist or that the mail has been flagged as SPAM.

Verify MIME type

The content of an attached file can be checked to see if it agrees with its stated filetype. A list of all filetypes that are verified in this way can be found in Appendix C, *Verified MIME filetypes*. This same option is also avaiable in the HTTP ALG and a fuller description of how it works can be found in Section 7.2.2, “The HTTP ALG”.

Block/Allow filetype

Filetypes from a predefined list can optionally be blocked or allowed as mail attachments and new filetypes can be added to the list. This same option is also avaiable in the HTTP ALG and a fuller description of how it works can be found in Section 7.2.2, “The HTTP ALG”. This same option is also available in the HTTP ALG and a fuller description of how it works can be found in Section 7.2.2, “The HTTP ALG”.

Anti-Virus Scanning

The CorePlus Anti-Virus subsystem can scan email attachments searching for malicious code. Suspect files can be dropped or just logged. This feature is common to a number of ALGs and is described fully in Section 7.4, “Anti-Virus Scanning”.

The Ordering for SMTP Filtering

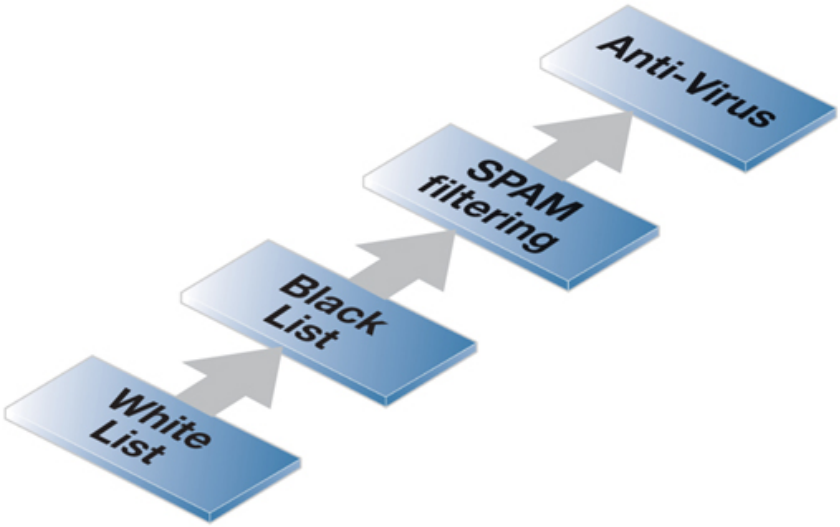
SMTP filtering obeys the following processing order and is similar to the order followed by the HTTP ALG except for the addition of SPAM filtering:

1. Whitelist.
2. Blacklist.
3. SPAM filtering (if enabled).
4. Anti-virus scanning (if enabled).

As described above, if an address is found on the whitelist then it will not be blocked if it also found on the blacklist. SPAM filtering, if it is enabled, is still applied to whitelisted addresses but emails flagged as SPAM will not be tagged nor dropped, only logged. Anti-virus scanning, if it is enabled, is always applied, even though an email's address is whitelisted.

Notice that either an email's sender or receiver address can be the basis for blocking by one of the first two filtering stages.

Figure 7.3. SMTP ALG Processing Order



Using Wildcards in White and Blacklists

Entries made in the white and blacklists can make use of *wildcarding* to have a single entry cover a large number of potential email addresses. The wildcard character "*" can be used to represent any sequence of characters.

For instance, the address entry **@some_domain.com* can be used to specify all possible email addresses for *some_domain.com*.

If, for example, wildcarding is used in the blacklist to block all addresses for a certain company called *my_company* then the blacklist address entry required could be **@my_company.com*.

If we want to now explicitly allow mails for just one department called *my_department* in *my_company* then this could be done with an entry in the whitelist of the form *my_department@my_company.com*.

7.2.5.1. DNSBL SPAM Filtering

Unsolicited email, often referred to as *SPAM*, has become both a major annoyance as well as a security issue on the public Internet. Unsolicited email, sent out in massive quantities by groups

known as *spammers*, can waste resources, transport malware as well as try to direct the reader to webpages which might exploit browser vulnerabilities.

Integral to the CorePlus SMTP ALG is a SPAM module that provides the ability to apply *spam filtering* to incoming email based on its origin. This can significantly reduce the burden of such email in the mailboxes of users behind the Clavister Security Gateway. CorePlus offers two approaches to handling SPAM:

- Dropping email which has a very high probability of being SPAM.
- Letting through but flagging email that has a moderate probability of being SPAM.

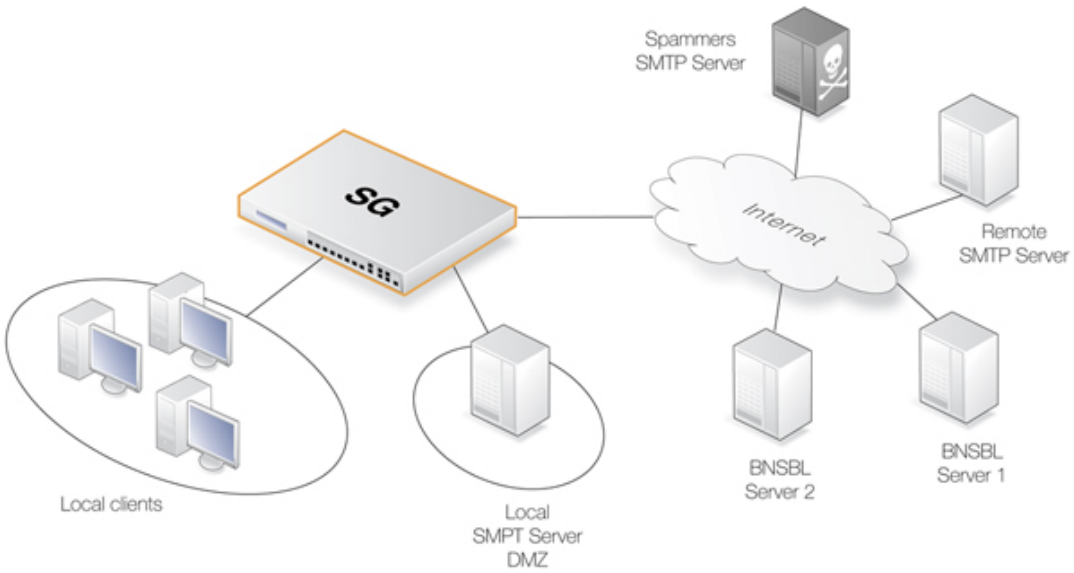
The CorePlus Implementation

SMTP functions as a protocol for sending emails between servers. CorePlus applies SPAM filtering to emails as they pass through the Clavister Security Gateway from a remote SMTP server to the local SMTP server (from which local clients will later download the emails). Typically the local SMTP server will be set up on a DMZ and there will usually be only one "hop" between the sending server and the local, receiving server.

A number of trusted organizations maintain publicly available databases of the origin IP address of known spamming SMTP servers and these can be queried over the public Internet. These lists are known as *DNS Black List* (DNSBL) databases and the information is accessible using a standardized query method supported by CorePlus. The image below illustrates all the components involved:

When the CorePlus SPAM filtering function is configured, the IP address of the email's sending server can be sent to one or more DNSBL servers to find out if any DNSBL servers think it is from a spammer or not (CorePlus examines the IP packet headers to do this). The reply sent back by a server is either a *not listed* response or a *listed* response. In the latter case of being listed, the DSNBL server is indicating the email might be SPAM and it will usually also provide information known as a *TXT* record which is a textual explanation for the listing.

Figure 7.4. DNSBL SPAM Filtering



The administrator can configure the CorePlus SMTP ALG to consult multiple DNSBL servers in order to form a consensus opinion on an email's origin address. As each new email arrives, servers are queried to assess the likelihood that the email is SPAM, based on its origin address. The CorePlus administrator assigns a weight greater than zero to each configured server so that a weighted sum can then be calculated based on all responses. The administrator can configure one of

the following actions based on the sum calculated:

1. **Dropped**
   If the sum is greater than or equal to a pre-defined *Drop threshold* then the email is considered to be definitely SPAM and is discarded or alternatively sent to a single, special mailbox.

   If it is discarded then the administrator has the option that an error message is sent back to the sending SMTP server (this error message is similar to the one used with blacklisting).

2. **Flagged as SPAM**
   If the sum is greater than or equal to a pre-defined *SPAM threshold* then the email is considered as probably being SPAM but forwarded to the recipient with notifying text inserted into it.

A Threshold Calculation Example

As an example, lets suppose that three DNSBL servers are configured: *dnsbl1*, *dnsbl2* and *dnsbl3*. Weights of **3**, **2** and **2** are assigned to these respectively. The SPAM threshold is then set to be **5**.

If *dnsbl1* and *dnsbl2* say an email is SPAM but *dnsbl3* does not, then the total calculated will be **3+2+0=5**. Since the total of **5** is equal to (or greater than) the threshold then the email will be treated as SPAM.

If the *Drop threshold* in this example is set at **7** then all three DNSBL servers would have to respond in order for the calculated sum to cause the email to be dropped (**3+2+2=7**).

Alternative Actions for Dropped SPAM

If the calculated sum is greater than or equal to the *Drop threshold* value then the email is not forwarded to the intended recipient. Instead the administrator can choose one of two alternatives for dropped email:

- A special email address can be configured to receive all dropped email. If this is done then any *TXT* messages sent by the DNSBL servers (described next) that identified the email as SPAM can be optionally inserted by CorePlus into the header of the forwarded email.

- If no receiver email address is configured for dropped emails then they are discarded by CorePlus. The administrator can specify that an error message is sent back to the sender address along with the *TXT* messages from the DNSBL servers that failed the email.

Tagging SPAM

If an email is considered to be probably SPAM because the calculated sum is above the SPAM threshold but it is below the Drop threshold, then the *Subject* field of the email is changed and pre-fixed with a message and the email is forwarded on to the intended recipient. The tag message text is specified by the administrator but can be left blank (although that is not recommended).

An example of tagging might be if the original *Subject* field is:

```
Buy this stock today!
```

And if the tag text is defined to be "***** SPAM *****", then the modified email's *Subject* field will become:

```
*** SPAM *** Buy this stock today!
```

And this is what the email's recipient will see in the summary of their inbox contents. The individual user could then decide to set up their own filters in the local client to deal with such tagged emails,

possibly sending it to a separate folder.

Adding X-SPAM Information

If an email is determined to be SPAM and a forwarding address is configured for dropped emails, then the administrator has the option to *Add TXT Records* to the email. A *TXT Record* is the information sent back from the DNSBL server when the server thinks the sender is a source of SPAM. This information can be inserted into the header of the email using the *X-SPAM* tagging convention before it is sent on. The X-SPAM fields added are:

- **X-Spam-Flag** - This value will always be *Yes*.
- **X-Spam-Checker-Version** - The CorePlus version that tagged the email.
- **X-Spam-Status** - This will always be *DNSBL*.
- **X-Spam-Report** - A list of DNSBL servers that flagged the email as SPAM.
- **X-Spam-TXT-Records** - A list of TXT records sent by the DNSBL servers that identified the email as SPAM.
- **X-Spam_Sender-IP** - IP address used by the email sender.

These fields can be referred to in filtering rules set up by the administrator in mail server software.

Allowing for Failed DNSBL Servers

If a query to a DNSBL server times out then CorePlus will consider that the query has failed and the weight given to that server will be automatically subtracted from both the SPAM and Drop thresholds for the scoring calculation done for that email.

If enough DNSBL servers do not respond then this subtraction could mean that the threshold values become negative. Since the scoring calculation will always produce a value of zero or greater (servers cannot have negative weights) then all email will be allowed through if both the SPAM and Drop thresholds become negative.

A log message is generated whenever a configured DNSBL server does not respond within the required time. This is done only once at the beginning of a consecutive sequence of response failures from a single server to avoid unnecessarily repeating the message.

Verifying the Sender Email

As part of the Anti-SPAM module, the option to verify the email sender denies emails with a mismatch of the SMTP "From" address and the header "From" address. In other words, the source address in the SMTP protocol header and the SMTP data load header must be the same. Spamming can cause these to be different so this feature provides an extra check on email integrity.

Logging

There are three types of logging done by the SPAM filtering module:

- Logging of dropped or SPAM tagged emails - These log messages include the source email address and IP as well as its weighted points score and which DNSBLs caused the event.
- DNSBLs not responding - DNSBL query timeouts are logged.
- All defined DNBSLs stop responding - This is a high severity event since all email will be allowed through if this happens.

Setup Summary

To set up DNSBL SPAM filtering in the SMTP ALG, the following list summarizes the steps:

- Specify which DNSBL servers are to be used. There can be multiple and they can act both as backups to each other as well as confirmation of a sender's status.
- Specify a *weight* for each server which will determine how important it is in deciding if email is SPAM or not in the calculation of a weighted sum.
- Specify the threshold for designating an email as SPAM. If the weighted sum is equal or greater than this then an email will be considered to be SPAM.
- Specify a textual tag to prefix to the **Subject** field of email designated as SPAM.
- Specify the *Drop threshold*. If the weighted sum is equal or greater than this then an email will be dropped entirely. This threshold should be greater or equal to the SPAM threshold. If they are equal then the Drop threshold will have precedence so that all email will be dropped when that threshold is reached.
- Optionally specify an email address to which dropped email will be sent (as an alternative to simply discarding it). Optionally specify that the *TXT* messages sent by the DNSBL servers that failed are inserted into the header of these emails.

Caching Addresses for Performance

To speed processing CorePlus maintains a cache of the most recently looked-up sender addresses in local memory. If the cache becomes full then the oldest entry is written over first.

The *Address Timeout* value for the cache can be changed by the administrator. This determines how long any address will be valid for once it is saved in the cache. After this period of time has expired, a new query for a cached sender address must be sent to the DNSBL servers.

The cache is emptied at startup or reconfiguration and its size of this cache can be controlled by the administrator.

PinPoint™ Real-time Monitoring

The following values for SPAM filtering can be monitored in real-time with the Clavister PinPoint product.

For the DNSBL subsystem overall:

- Number of emails checked.
- Number of emails SPAM tagged.
- Number of dropped emails.

For each DNSBL server accessed:

- Number of positive (is SPAM) responses from each configured DNSBL server.
- Number of queries sent to each configured DNSBL server.
- Number of failed queries (without replies) for each configured DNSBL server.

The *dnsbl* CLI Command

The **dnsbl** CLI command provides a means to control and monitor the operation of the SPAM filtering module. The **dnsbl** command on its own without options shows the overall status of all ALGs. If the name of the SMTP ALG object on which DNSBL SPAM filtering is enabled is *my_smtp_alg* then the output would be:

```
Device:/> dnsbl

DNSBL Contexts:

Name                        Status      Spam     Drop    Accept
--------------------------  -------- -------- -------- --------
my_smtp_alg                   active      156       65    34299
alt_smtp_alg                inactive        0        0        0
```

The *-show* option provides a summary of the SPAM filtering operation of a specific ALG. It is used below to examine activity for *my_smtp_alg* although in this case, the ALG object has not yet processed any emails.

```
Device:/> dnsbl my_smtp_alg -show

Drop Threshold        : 20
Spam Threshold        : 10
Use TXT records       : yes
IP Cache disabled
Configured BlackLists : 4
Disabled BlackLists   : 0
Current Sessions      : 0
Statistics:
Total number of mails checked : 0
Number of mails dropped       : 0
Number of mails spam tagged   : 0
Number of mails accepted      : 0
BlackList                 Status   Value Total    Matches  Failed
------------------------- -------- ----- -------- -------- --------
zen.spamhaus.org          active      25        0        0        0
cbl.abuseat.org           active      20        0        0        0
dnsbl.sorbs.net           active       5        0        0        0
asdf.egrhb.net            active       5        0        0        0
```

To examine the statistics for a particular DNSBL server, the following command can be used.

```
Device:/> dnsbl smtp_test zen.spamhaus.org -show

BlackList: zen.spamhaus.org
Status       : active
Weight value : 25
Number of mails checked                    : 56
Number of matches in list                  : 3
Number of failed checks (times disabled)   : 0
```

To clean out the **dnsbl** cache for *my_smtp_alg* and to reset all its statistical counters, the following command option can be used:

```
Device:/> dnsbl my_smtp_alg -clean
```

Note
A list of DNSBL servers can be found at http://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists.

7.2.6. The POP3 ALG

POP3 is a mail transfer protocol that differs from SMTP in that the transfer of mail is directly from a server to a user's client software.

POP3 ALG Options

Key features of the POP3 ALG are:

Block Clear Text Authentication — Block connections between client and server that send the username/password combination as clear text which can be easily read (some servers may not support other methods than this).

Hide User — This option prevents the POP3 server from revealing that a username does not exist. This prevents users from trying different usernames until they find a valid one.

Allow Unknown Commands — Non-standard POP3 commands not recognized by the ALG can be allowed or disallowed.

Fail Mode — When content scanning find bad file integrity then the file can be allowed or disallowed.

Verify MIME type — The content of an attached file can be checked to see if it agrees with its stated filetype. A list of all filetypes that are verified in this way can be found in Appendix C, *Verified MIME filetypes*. This same option is also avaiable in the HTTP ALG and a fuller description of how it works can be found in Section 7.2.2, “The HTTP ALG”.

Block/Allow filetype — Filetypes from a predefined list can optionally be blocked or allowed as mail attachments and new filetypes can be added to the list. This same option is also avaiable in the HTTP ALG and a fuller description of how it works can be found in Section 7.2.2, “The HTTP ALG”.

Anti-Virus Scanning — The CorePlus Anti-Virus subsystem can optionally scan email attachments searching for malicious code. Suspect files can be dropped or just logged. This feature is common to a number of ALGs and is described fully in Section 7.4, “Anti-Virus Scanning”.

7.2.7. The SIP ALG

Session Initiation Protocol (SIP) is an ASCII (UTF-8) text based signalling protocol used to establish sessions between peers in an IP network. It is a request-response protocol that resembles HTTP and SMTP. A session might consist of a Voice-Over-IP (VoIP) telephone call or it could be a collaborative multi-media conference. Using SIP with VoIP means that telephony can become another IP application which can integrate into other services.

SIP does not know about the details of a session's content and is only responsible for initiating, terminating and modifying sessions. Sessions set up by SIP are typically used for the streaming of

audio and video over the Internet using the UDP protocol but they might also involve traffic based on the TCP protocol. Although UDP based VoIP sessions are a common use, communication using other protocols such as TCP or TLS might be involved in a session.

SIP is defined by the IETF standard RFC 3261 and is becoming popular as the standard for VoIP. It is comparable to H.323 but a design goal with SIP is to make it more scalable that H.323. (For VoIP see also Section 7.2.9, “The H.323 ALG”.)

SIP Components

The following components are the logical building blocks for SIP communication:

User Agents These are the end points or "peers" that are involved in the peer-to-peer communication. These would typically be the workstation or device used in an IP telephony conversation. The word *peer* will often be used in this section in this context.

Proxy Servers These act as routers in the SIP protocol, performing both as peer and server when receiving peer requests. They forward requests to a peer's current location as well as authenticating and authorizing access to services. They also implement provider call-routing policies.

The proxy is typically located on the unprotected side of the Clavister Security Gateway and this is the proxy location supported by the CorePlus SIP ALG.

Registrars A server that handles SIP REGISTER requests is given the special name of Registrar. The Registrar server has the task of locating the host where the other peer is reachable.

The Registrar and Proxy Server are logical entities and my in fact reside in the same physical server.

SIP Media-related Protocols

SIP sessions make use of a number of sub-protocols:

SDP *Session Description Protocol* (RFC4566) is used for media session initialization.

RTP *Real-time Transport Protocol* (RFC3550) is used as the underlying packet format for delivering audio and video streaming via IP using the UDP protocol.

RTCP *Real-time Control Protocol* (RFC3550) is used in conjunction with RTP to provide out-of-band control flow management.

SIP Usage Scenarios

The CorePlus SIP ALG supports the following usage scenarios:

1. Internal to External The SIP session is between a peer on the protected side of a Clavister Security Gateway and a peer which is on the external, unprotected side. Communication typically takes place across the public Internet.

2. Same Network A refinement of the internal to internal scenario is the case where the two peers in a session reside on the same network.

In all these scenarios, the proxy server is assumed to be on the unprotected side of the Clavister Security Gateway.

SIP Configuration Options

The following options can be configured for a SIP ALG object:

Maximum Sessions per ID — The number of simultaneous sessions that a single peer can be involved with is restricted by this value. The default number is *5*.

Maximum Registration Time — The maximum time for registration with a SIP Registrar. The default value is *3600* seconds.

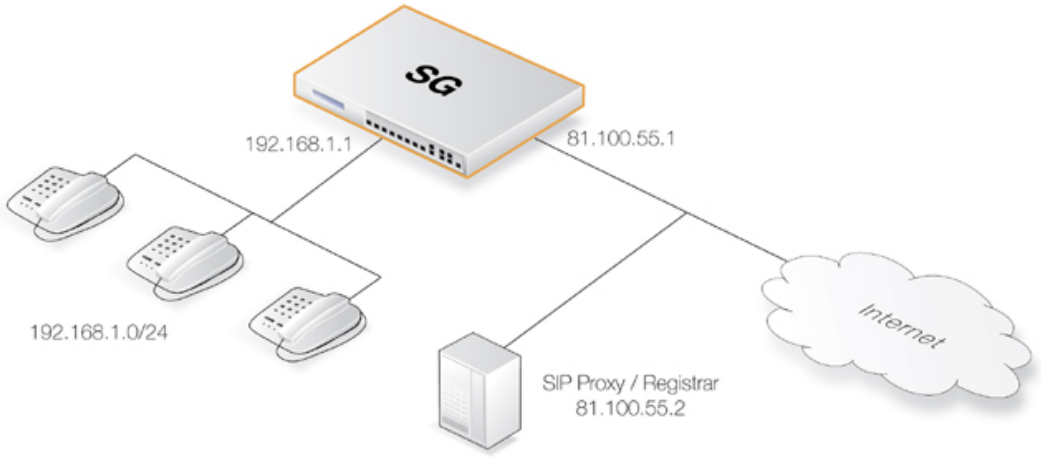
SIP Request-Response Timeout — The maximum time allowed for responses to SIP requests. A timeout condition occurs after this wait. The default is *180 seconds*.

SIP Signal Timeout — The maximum time allowed for SIP sessions. The default value is *43200 seconds*.

Data Channel Timeout — The maximum time allowed for periods with no traffic in a SIP session. A timeout condition occurs if this value is exceeded. The default value is *120 seconds*

SIP Setup Summary

For setup we will assume a scenario where there is an office with VoIP users on a private internal network and the network's topology will be hidden using NAT. This scenario is illustrated below.



The SIP proxy in the above diagram could alternatively be located remotely across the Internet. The SIP proxy server should be configured with the feature **Record-Route Enabled** to insure all SIP traffic to and from the office peers will be sent through the SIP Proxy. This is recommended since the attack surface is minimimized by allowing only SIP signalling from the SIP Proxy to enter the local network. The steps to follow are:

Note

SIP User Agents and SIP Proxies should not be configured to employ NAT Traversal in a setup. For instance the Simple Traversal of UDP through NATs (STUN) technique should not be used. The CorePlus SIP ALG will take care of all traversal issues with NAT in a SIP setup.

1. Define a *SIP ALG* object using the options described above.

2. A *Service* object is used for the ALG which has the above SIP ALG associated with it. The Service should have:

- **Destination Port** set to *5060*.
- **Type** set to *UDP*.

3. Define two rules in the IP rule set:

   - A **NAT** rule for outbound traffic from user agents on the internal network to the SIP Proxy Server located externally. The SIP ALG will take care of all address translation needed by the NAT rule. This translation will occur both on the IP level and the application level. Neither the user agents or the proxies need to be aware that the local users are being NATed.
   - An **Allow** rule for inbound SIP traffic from the SIP proxy to the IP of the Clavister Security Gateway. This rule will use **core** (in other words CorePlus itself) as the destination interface. The reason for this is due to the NAT rule above. When an incoming call is received, CorePlus will automatically locate the local receiver, perform address translation and forward SIP messages to the receiver. This will be executed based on the ALGs internal state.

   A **SAT** rule for translating incoming SIP messages is not needed since the ALG will automatically redirect incoming SIP requests to the correct internal user. When a SIP user agent behind a NATing Clavister Security Gateway registers with an external SIP proxy, CorePlus sends its own IP address as contact information to the SIP proxy. CorePlus registers the user-agent's local contact information and uses this to redirect incoming requests to the user. The ALG takes care of the address translations needed.

4. Ensure the peers are correctly configured. The SIP Proxy Server plays a key role in locating the current location of the other peer for the session. The proxy's IP address is not specified directly in the ALG. Instead its location is either entered directly into the client software used by the peer or in some cases the peer will have a way of retrieving the proxy's IP address automatically such as through DHCP.

Handling Data Traffic

The setup steps above take care of the SIP communication for establishing peer-to-peer communications. The two IP rules are always needed so that peers can access the SIP proxy but no rules are needed to handle the actual data traffic involved in, for example, a VoIP call. The SIP ALG automatically takes care of establishing the CorePlus objects required for allowing the data traffic to traverse the Clavister Security Gateway and these are invisible to the administrator.

Tip

Make sure there are no preceding rules already in the IP rule set disallowing or allowing the same kind of traffic.

Depending on the SIP environment, the CorePlus SIP ALG can operate in hidden-topology environments with private IP addresses, as well as open environments with public IP addresses. SIP is a highly configurable protocol and the following describes the configuration required.

SIP and Virtual Routing

SIP is a complex protocol that requires that CorePlus maintains state information regarding active calls and registered users. The scope of the SIP state is uniquely defined by the SIP-service and its connected SIP-ALG. If multiple SIP domains are running concurrently but using different Policy-based Routing Tables then one SIP-ALG and one SIP-service object should be created for each table. Policy-based Routing Tables are described in detail in Section 5.4, “Virtual Routing”.

Example 7.4. External SIP Proxy with Hidden Clients

This example show how to implement the scenario described above.

The peer is assumed to be on the network *if1_net* connected to the interface *if1*. The SIP proxy is assumed to be on the IP address *proxy_ip* on the interface *ext*.

Web Interface

A. Define the following IP objects:

- **if1_net:** 192.168.1.0/24
  (the internal network)
- **proxy_ip:** 81.100.55.2
  (the SIP proxy)
- **ip_wan:** 81.100.55.1
  (the Clavister Security Gateway's public IP address)

B. Define an SIP ALG object

1. Go to **Objects > ALG > Add > SIP ALG**
2. The **SIP ALG** properties will be displayed
3. Specify a name for the ALG, for example *sip_alg*
4. Click **OK**

C. Define a custom *Service* object for SIP:

1. Go to **Objects > Services > Add > TCP/UDP**
2. The **Service** properties will be displayed
3. Specify a name for the service, for example *sip_serv*
4. Choose *UDP* as the **Type**
5. Choose *sip-alg* as the **ALG**
6. Under **Destination** and enter port number *5060*
7. Click **OK**

D. Define the outgoing SIP traffic IP rule:

1. Go to **Rules > IP Rule Set > main > Add > IP Rule**
2. The **Rule Properties** dialog will be displayed
3. Now enter:
   - **Name:** sip_nat
   - **Action:** NAT
   - **Service:***sip_serv*
   - **Source Interface:** if1
   - **Source Network:** if1_net
   - **Destination Interface:** ext
   - **Destination Network:** proxy_ip
   - **Comment:** Allow outgoing SIP calls
4. Click **OK**

E. Define the incoming SIP traffic IP rule:

1. Go to **Go to Rules > IP Rule Set > main > Add > IP Rule**
2. The **Rule Properties** dialog will be displayed
3. Now enter:
   - **Name:** sip_allow
   - **Action:** Allow
   - **Service:** *sip_serv*
   - **Source Interface:** ext
   - **Source Network:** proxy_ip
   - **Destination Interface:** core
   - **Destination Network:** ip_wan
   - **Comment:** Allow incoming SIP traffic
4. Click **OK**

7.2.8. The SIP ALG

Session Initiation Protocol (SIP) is an ASCII (UTF-8) text based signalling protocol used to establish sessions between clients in an IP network. It is a request-response protocol that resembles HTTP and SMTP. The session which SIP sets up might consist of a Voice-Over-IP (VoIP) telephone call or it could be a collaborative multi-media conference. Using SIP with VoIP means that telephony can become another IP application which can integrate into other services.

SIP does not know about the details of a session's content and is only responsible for initiating, terminating and modifying sessions. Sessions set up by SIP are typically used for the streaming of audio and video over the Internet using the RTP/RTCP protocol (which is based on UDP) but they might also involve traffic based on the TCP protocol. A RTP/RTCP based sessions might also involve TCP or TLS based traffic in the same session.

SIP is defined by IETF RFC 3261 and is considered an important standard for VoIP communication. It is comparable to H.323 but a design goal with SIP was to make it more scalable than H.323. (For VoIP see also Section 7.2.9, “The H.323 ALG”.)

SIP Components

The following components are the logical building blocks for SIP communication:

User Agents These are the end points or *clients* that are involved in the client-to-client communication. These would typically be the workstation or device used in an IP telephony conversation. The term *client* will be used throughout this section to describe a *user agent*.

Proxy Servers These act as routers in the SIP protocol, performing both as client and server when receiving client requests. They forward requests to a client's current location as well as authenticating and authorizing access to services. They also implement provider call-routing policies.

The proxy is often located on the external, unprotected side of the Clavister Security Gateway but can have other locations. All of these scenarios are supported by CorePlus.

Registrars A server that handles SIP *REGISTER* requests is given the special name of Registrar. The Registrar server has the task of locating the host where the

other client is reachable.

The Registrar and Proxy Server are logical entities and may, in fact, reside on the same physical server.

SIP Media-related Protocols

A SIP session makes use of a number of protocols. These are:

SDP *Session Description Protocol* (RFC4566) is used for media session initialization.

RTP *Real-time Transport Protocol* (RFC3550) is used as the underlying packet format for delivering audio and video streaming via IP using the UDP protocol.

RTCP *Real-time Control Protocol* (RFC3550) is used in conjunction with RTP to provide out-of-band control flow management.

CorePlus SIP Setup

When configuring CorePlus to handle SIP sessions the following steps are needed:

- Define a single *Service* object for SIP communication.
- Define a *SIP ALG* object which is associated with the *Service* object.
- Define the appropriate *IP rules* for SIP communications which use the defined *Service* object.

SIP ALG Options

The following options can be configured for a SIP ALG object:

Maximum Sessions per ID The number of simultaneous sessions that a single client can be involved with is restricted by this value. The default number is *5*.

Maximum Registration Time The maximum time for registration with a SIP Registrar. The default value is *3600* seconds.

SIP Signal Timeout The maximum time allowed for SIP sessions. The default value is *43200 seconds*.

Data Channel Timeout The maximum time allowed for periods with no traffic in a SIP session. A timeout condition occurs if this value is exceeded. The default value is *120 seconds*.

Allow Media Bypass If this option is enabled then data (e.g RTP/RTCP) communication may take place directly between two clients without involving the Clavister Security Gateway. This would only happen if the two clients were behind the same interface and belong to the same network. The default value is *Disabled*.

The SIP Proxy *Record-Route* Option

To understand how to set up SIP scenarios with CorePlus, it is important to first understand the SIP

proxy *Record-Route* option. SIP proxies have the Record-Route option either enabled or disabled. When it is switched on, a proxy is known as a *Stateful proxy*. When Record-Route is enabled, a proxy is saying it will be the intermediary for all SIP signalling that takes place between two clients.

When a SIP session is being set up, the calling client sends an *INVITE* message to its outbound SIP proxy server. The SIP proxy relays this message to the remote proxy server responsible for the called, remote client's contact information. The remote proxy then relays the *INVITE* message to the called client. Once the two clients have learnt of each other's IP addresses, they can communicate directly with each other and remaining SIP messages can bypass the proxies. This facilitates scaling since proxies are used only for the initial SIP message exchange.

The disadvantage of removing proxies from the session is that CorePlus IP rules must be set up to allow all SIP messages through the Clavister Security Gateway, and if the source network of the messages is not known then a large number of potentially dangerous connections must be allowed by the IP rule set. This problem does not occur if the local proxy is set up with the Record-Route option enabled. In this mode, all SIP messages will only come from the proxy.

The different rules required when the *Record-Route* option is enabled and disabled can be seen in the two different sets of IP rules listed below in the detailed description of *Scenario 1 Protecting local clients - Proxy located on the Internet*.

IP Rules for Media Data

When discussing SIP there are two distinct exchanges under discussion:

- The SIP session which sets up communication between two clients prior to the exchange of *media data*.
- The exchange of the *media data* itself, for example the coded voice data which constitute a VoIP phone call.

In the SIP setup described below, the IP rules described must be explicitly defined to deal with the first of these: the SIP exchanges needed for establishing client-to-client communications. However no IP rules or other objects need to be defined to handle the exchange of media data. The SIP ALG automatically and invisibly takes care of creating the connections required (sometimes described as *pinholes*) for allowing the media data traffic to traverse the Clavister Security Gateway.

Tip
Make sure there are no preceding rules already in the IP rule set disallowing or allowing the same kind of traffic.

SIP and Virtual Routing

SIP is a complex protocol that requires that CorePlus maintains state information regarding active calls and registered users. The scope of a SIP state is uniquely defined by the SIP-service and its connected SIP ALG. If multiple SIP domains are running concurrently but using different Policy-based Routing Tables then one SIP ALG and one SIP Service object should be created for each table. Policy-based Routing Tables are described in detail in Section 5.4, “Virtual Routing”.

SIP Usage Scenarios

CorePlus supports a variety of SIP usage scenarios. The following three scenarios cover nearly all possible types of usage:

- **Scenario 1**
  Protecting local clients - Proxy located on the Internet

  The SIP session is between a client on the local, protected side of the Clavister Security

Gateway and a client which is on the external, unprotected side. The SIP proxy is located on the external, unprotected side of the Clavister Security Gateway. Communication typically takes place across the public Internet with clients on the intertnal, protected side registering with a proxy on the public, unprotected side.

- **Scenario 2**
  Protecting proxy and local clients - Proxy on the same network as clients

  The SIP session is between a client on the local, protected side of the Clavister Security Gateway and a client which is on the external, unprotected side. The SIP proxy is located on the local, protected side of the Clavister Security Gateway and can handle registrations from both clients located on the same local network as well as clients on the external, unprotected side. Communication can take place across the public Internet or between clients on the local network.

- **Scenario 3**
  Protecting proxy and local clients - Proxy on a DMZ interface

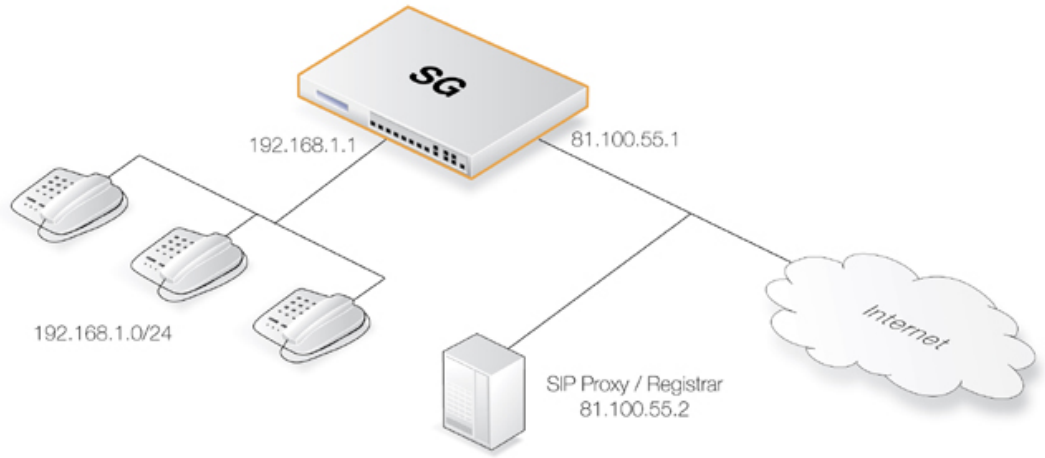
  The SIP session is between a client on the local, protected side of the Clavister Security Gateway and a client which is on the external, unprotected side. The SIP proxy is located on the DMZ interface and is physically separated from the local client network as well as the remote client network and proxy network.

All the above scenarios will also deal with the situation where two clients in a session reside on the same network.

These scenarios will now be examined in detail.

Scenario 1
Protecting local clients - Proxy located on the Internet

The scenario assumed is an office with VoIP users on a private internal network where the network's topology will be hidden using NAT. This is illustrated below.



The SIP proxy in the above diagram could alternatively be located remotely across the Internet. The proxy should be configured with the **Record-Route** feature enabled to insure all SIP traffic to and from the office clients will be sent through the SIP Proxy. This is recommended since the attack surface is minimized by allowing only SIP signalling from the SIP Proxy to enter the local network.

This scenario can be implemented in two ways:

- Using NAT to hide the network topology.
- Without NAT so the network topology is exposed.

Note

SIP User Agents and SIP Proxies should not be configured to employ NAT Traversal in a setup. For instance the Simple Traversal of UDP through NATs (STUN) technique should not be used. The CorePlus SIP ALG will take care of all NAT traversal issues in a SIP scenario.

The setup steps for this scenario are as follows:

1. Define a *SIP ALG* object using the options described above.
2. Define a *Service* object which is associated with the SIP ALG object. The Service should have:
   - **Destination Port** set to *5060* (the default SIP signalling port).
   - **Type** set to *TCP/UDP*.
3. Define two rules in the IP rule set:
   - A **NAT** rule for outbound traffic from clients on the internal network to the SIP Proxy Server located externally. The SIP ALG will take care of all address translation needed by the NAT rule. This translation will occur both on the IP level and the application level. Neither the clients or the proxies need to be aware that the local users are being NATed.
   - An **Allow** rule for inbound SIP traffic from the SIP proxy to the IP of the Clavister Security Gateway. This rule will use **core** (in other words, CorePlus itself) as the destination interface. The reason for this is due to the NAT rule above. When an incoming call is received, CorePlus will automatically locate the local receiver, perform address translation and forward SIP messages to the receiver. This will be executed based on the ALGs internal state.

   A **SAT** rule for translating incoming SIP messages is not needed since the ALG will automatically redirect incoming SIP requests to the correct internal user. When a SIP client behind a NATing Clavister Security Gateway registers with an external SIP proxy, CorePlus sends its own IP address as contact information to the SIP proxy. CorePlus registers the client's local contact information and uses this to redirect incoming requests to the user. The ALG takes care of the address translations needed.
4. Ensure the clients are correctly configured. The SIP Proxy Server plays a key role in locating the current location of the other client for the session. The proxy's IP address is not specified directly in the ALG. Instead its location is either entered directly into the client software used by the client or in some cases the client will have a way of retrieving the proxy's IP address automatically such as through DHCP.

Note

SIP User Agents and SIP Proxies should not be configured to employ NAT Traversal in a setup. For instance the Simple Traversal of UDP through NATs (STUN) technique should not be used. The CorePlus SIP ALG will take care of all traversal issues with NAT in a SIP setup.

The IP rules with the Record-Route option enabled would be as shown below, the changes that apply when NAT is used are shown in parentheses "*(..)*".

Action	Src Interface	Src Network	Dest Interface	Dest Network
Allow (or NAT)	lan	lannet	wan	ip_proxy
Allow	wan	ip_proxy	lan	lannet

Action	Src Interface	Src Network	Dest Interface	Dest Network
			(or core)	(or ip_wan)

Without the Record-Route option enabled the IP rules would be as shown below, the changes that apply when NAT is used are again shown in parentheses "*(..)*".

Action	Src Interface	Src Network	Dest Interface	Dest Network
Allow (or NAT)	lan	lannet	wan	<All possible IPs>
Allow	wan	<All possible IPs>	lan (or core)	lannet (or ipwan)

The advantage of using *Record-Route* is clear since now the destination network for outgoing traffic and the source network for incoming traffic have to include all IP addresses that are possible.

The Service object for IP rules.
*In this section, tables which list IP rules like those above, will omit the **Service** object associated with the rule. The same, custom **Service** object is used for all SIP scenarios.*

Example 7.5. SIP with Local Clients, Proxy on Internet

This example shows the exact steps to implement **Scenario 1** which is described above. The local network topology is hidden using NAT. The proxy server lies on the external, unprotected side of the Clavister Security Gateway.

The client is assumed to be on the network *if1_net* connected to the interface *if1*. The SIP proxy is assumed to be on the IP address *proxy_ip* on the interface *ext*.

Web Interface

A. Define the following IP objects:

- **if1_net:** 192.168.1.0/24
  (the internal network)
- **proxy_ip:** 81.100.55.2
  (the SIP proxy)
- **ip_wan:** 81.100.55.1
  (the Clavister Security Gateway's public IP address)

B. Define an SIP ALG object

1. Go to **Objects > ALG > Add > SIP ALG**
2. The **SIP ALG** properties will be displayed
3. Specify a name for the ALG, for example *sip_alg*
4. Click **OK**

C. Define a custom *Service* object for SIP:

1. Go to **Objects > Services > Add > TCP/UDP**
2. The **Service** properties will be displayed
3. Specify a name for the service, for example *sip_serv*
4. Choose *UDP* as the **Type**
5. Choose *sip-alg* as the **ALG**
6. Under **Destination** and enter port number *5060*

7. Click **OK**

D. Define the outgoing SIP traffic IP rule:

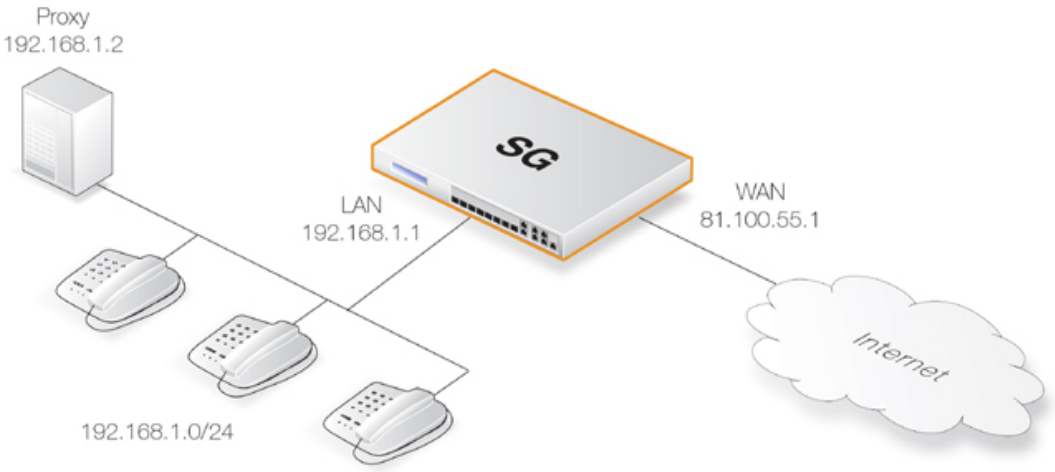
1. Go to **Rules > IP Rule Set > main > Add > IP Rule**
2. The **Rule Properties** dialog will be displayed
3. Now enter:
   - **Name:** sip_nat
   - **Action:** NAT
   - **Service:** sip_serv
   - **Source Interface:** if1
   - **Source Network:** if1_net
   - **Destination Interface:** ext
   - **Destination Network:** proxy_ip
   - **Comment:** Allow outgoing SIP calls
4. Click **OK**

E. Define the incoming SIP traffic IP rule:

1. Go to **Rules > IP Rule Set > main > Add > IP Rule**
2. The **Rule Properties** dialog will be displayed
3. Now enter:
   - **Name:** sip_allow
   - **Action:** Allow
   - **Service:** sip_serv
   - **Source Interface:** ext
   - **Source Network:** proxy_ip
   - **Destination Interface:** core
   - **Destination Network:** ip_wan
   - **Comment:** Allow incoming SIP traffic
4. Click **OK**

Scenario 2
Protecting proxy and local clients - Proxy on the same network as clients

In this scenario the goal is to protect the local clients as well as the SIP proxy. The proxy is located on the same, local network as the clients, with SIP signalling and media data flowing across two interfaces. This scenario is illustrated below.



This scenario can be implemented in two ways:

- Using NAT to hide the network topology.
- Without NAT so the network topology is exposed.

Solution A - Using NAT

Here, the proxy and the local clients are hidden behind the IP address of the Clavister Security Gateway. The setup steps are as follows:

1. Define a single SIP ALG object using the options described above.
2. Define a *Service* object which is associated with the SIP ALG object. The Service should have:
   - **Destination Port** set to *5060* (the default SIP signalling port)
   - **Type** set to *TCP/UDP*
3. Define three rules in the IP rule set:
   - A **NAT** rule for outbound traffic from the local proxy and the clients on the internal network to the remote clients on, for example, the Internet. The SIP ALG will take care of all address translation needed by the NAT rule. This translation will occur both on the IP level and the application level. Neither the clients or the proxies need to be aware that the local clients are being NATed.

     If *Record-Route* is enabled on the SIP proxy, the *source* network of the NAT rule can include only the SIP proxy, and not the local clients.
   - A **SAT** rule for redirecting inbound SIP traffic to the private IP address of the NATed local proxy. This rule will have **core** as the destination interface (in other words CorePlus itself) since inbound traffic will be sent to the private IP address of the SIP proxy.
   - An **Allow** rule which matches the same type of traffic as the **SAT** rule defined in the previous step.

	Action	Src Interface	Src Network	Dest Interface	Dest Network
OutboundFrom ProxyUsers	NAT	lan	lannet (ip_proxy)	wan	all-nets
InboundTo ProxyAndClients	SAT SETDEST ip_proxy	wan	all-nets	core	ip_wan
InboundTo ProxyAndClients	Allow	wan	all-nets	core	ip_wan

If *Record-Route* is enabled then the *Source Network* for outbound traffic from proxy users can be further restricted in the above rules by using "*ip_proxy*" as indicated.

When an incoming call is received, the SIP ALG will follow the **SAT** rule and forward the SIP request to the proxy server. The proxy will in turn, forward the request to its final destination which is the client.

If *Record-Route* is disabled at the proxy server, and depending on the state of the SIP session, the SIP ALG may forward inbound SIP messages directly to the client, bypassing the SIP proxy. This will happen automatically without further configuration.

Solution B - Without NAT

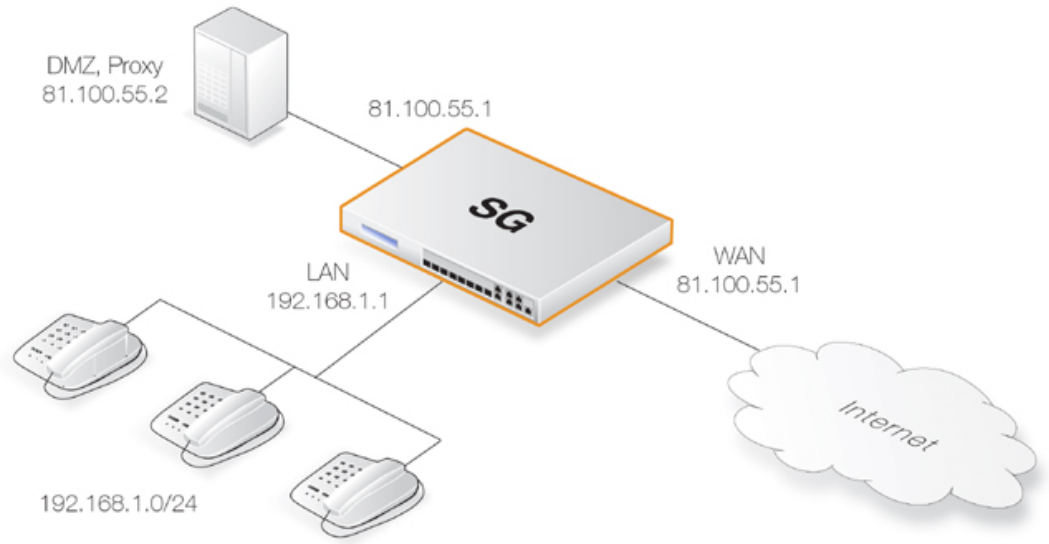
Without NAT, the outbound NAT rule is replaced by an **Allow** rule. The inbound **SAT** and **Allow** rules are replaced by a single **Allow** rule.

	Action	Src Interface	Src Network	Dest Interface	Dest Network
OutboundFrom Proxy&Clients	Allow	lan	lannet (ip_proxy)	wan	all-nets
InboundTo Proxy&Clients	Allow	wan	all-nets	lan	lannet (ip_proxy)

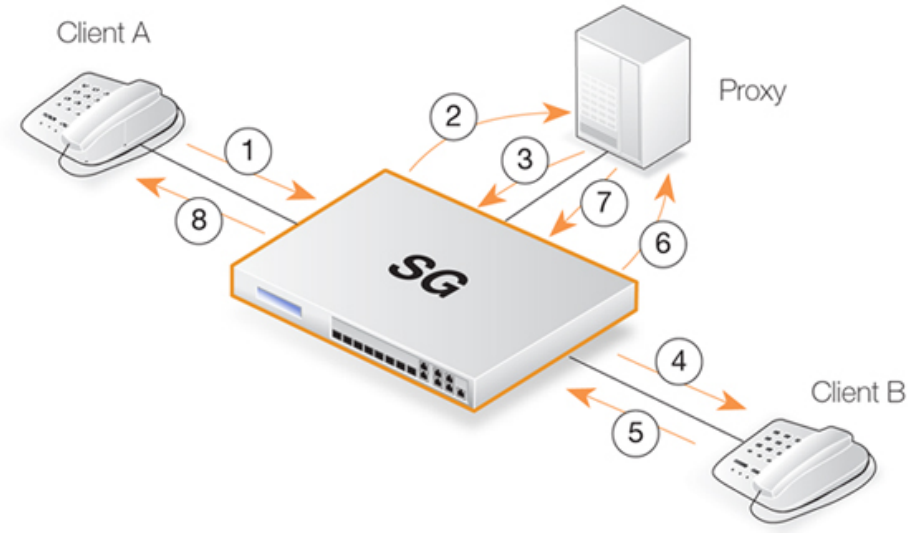
If *Record-Route* is enabled then the networks in the above rules can can be further restricted by using "*(ip_proxy)*" as indicated.

Scenario 3
Protecting proxy and local clients - Proxy on the DMZ interface

This scenario is similar to the previous but the major difference is the location of the local SIP proxy server. The server is placed on a separate interface and network to the local clients. This setup adds an extra layer of security since the initial SIP traffic is never exchanged directly between a remote endpoint and the local, protected clients.



The complexity is increased in this scenario since SIP messages flow across three interfaces: the receiving interface from the call initiator, the DMZ interface towards the proxy and the destination interface towards the call terminator. This the initial messages exchanges that take place when a call is setup in this scenario are illustrated below:



The exchanges illustrated are as follows:

- **1,2** - An initial *INVITE* is sent to the outbound local proxy server on the DMZ.
- **3,4** - The proxy server sends the SIP messages towards the destination on the Internet.
- **5,6** - A remote client or proxy server replies to the local proxy server.
- **7,8** - The local proxy forwards the reply to the local client.

This scenario can be implemented in a topology hiding setup with DMZ (**Solution A** below) as well as a setup without NAT (**Solution B** below).

Solution A - Using NAT

The following should be noted about this setup:

- The IP address of the SIP proxy must be a globally routable IP address. The Clavister Security Gateway does not support hiding of the proxy on the DMZ.
- The IP address of the DMZ interface must be a globally routable IP address. This address can be the same address as the one used on the external interface.

The setup steps are as follows:

1. Define a single SIP ALG object using the options described above.
2. Define a *Service* object which is associated with the SIP ALG object. The Service should have:
   - **Destination Port** set to *5060* (the default SIP signalling port)
   - **Type** set to *TCP/UDP*
3. Define four rules in the IP rule set:
   - A **NAT** rule for outbound traffic from the clients on the internal network to the proxy located on the DMZ interface. The SIP ALG will take care of all address translation needed by the NAT rule. This translation will occur both at the IP level and at the application level.

Note

Clients registering with the proxy on the DMZ will have the IP address of the DMZ interface as the contact address.

- An **Allow** rule for outbound traffic from the proxy behind the DMZ interface to the remote clients on the Internet.
- An **Allow** rule for inbound SIP traffic from the SIP proxy behind the DMZ interface to the IP address of the Clavister Security Gateway. This rule will have **core** (in other words, CorePlus itself) as the destination interface.

  The reason for this is because of the **NAT** rule above. When an incoming call is received, CorePlus automatically locates the local receiver, performs address translation and forwards SIP messages to the receiver. This is done based on the SIP ALG's internal state.
- An **Allow** rule for inbound traffic from, for example the Internet, to the proxy behind the DMZ.

4. If *Record-Route* is **not** enabled at the proxy, direct exchange of SIP messages must also be allowed between clients, bypassing the proxy. The following additional rules are therefore needed when *Record-Route* is disabled:

   - A **NAT** rule for outbound traffic from the clients on the internal network to the extrenal clients and proxies on, for example, the Internet. The SIP ALG will take care of all address translation needed by the NAT rule. The translation will occur both at the IP level and the application level.
   - An **Allow** rule for inbound SIP traffic from, for example the Internet, to the IP address of the DMZ interface. The reason for this is because local clients will be NATed using the IP address of the DMZ interface when they register with the proxy located on the DMZ.

     This rule has **core** as the destination interface (in other words, CorePlus itself). When an incoming call is received, CorePlus uses the registration information of the local receiver to automatically locate this receiver, perform address translation and forward SIP messages to the receiver. This will be done based on the internal state of the SIP ALG.

The IP rules needed with *Record-Route* enabled are:

	Action	Src Interface	Src Network	Dest Interface	Dest Network
OutboundToProxy	NAT	lan	lannet	dmz	ip_proxy
OutboundFromProxy	Allow	dmz	ip_proxy	wan	all-nets
InboundFromProxy	Allow	dmz	ip_proxy	core	ip_dmz
InboundToProxy	Allow	wan	all-nets	dmz	ip_proxy

With *Record-Route* disabled, the following IP rules must be added to those above:

	Action	Src Interface	Src Network	Dest Interface	Dest Network
OutboundBypassProxy	NAT	lan	lannet	wan	all-nets
InboundBypassProxy	Allow	wan	all-nets	core	ipdmz

Solution B - Without NAT

The setup steps are as follows:

1. Define a single SIP ALG object using the options described above.
2. Define a *Service* object which is associated with the SIP ALG object. The Service should have:
   - **Destination Port** set to *5060* (the default SIP signalling port)
   - **Type** set to *TCP/UDP*
3. Define four rules in the IP rule set:

- An **Allow** rule for outbound traffic from the clients on the internal network to the proxy located on the DMZ interface.
- An **Allow** rule for outbound traffic from the proxy behind the DMZ interface to the remote clients on the Internet.
- An **Allow** rule for inbound SIP traffic from the SIP proxy behind the DMZ interface to the clients located on the local, protected network.
- An **Allow** rule for inbound SIP traffic from clients and proxies on the Internet to the proxy behind the DMZ interface.

4. If *Record-Route* is **not** enabled at the proxy, direct exchange of SIP messages must also be allowed between clients, bypassing the proxy. The following two additional rules are therefore needed when *Record-Route* is disabled:
   - An **Allow** rule for outbound traffic from the clients on the local network to the external clients and proxies on the Internet.
   - An **Allow** rule for inbound SIP traffic from the Internet to clients on the local network.

The IP rules with *Record-Route* enabled are:

	Action	Src Interface	Src Network	Dest Interface	Dest Network
OutboundToProxy	Allow	lan	lannet	dmz	ip_proxy
OutboundFromProxy	Allow	dmz	ip_proxy	lan	lannet
InboundFromProxy	Allow	dmz	ip_proxy	core	ip_dmz
InboundToProxy	Allow	wan	all-nets	dmz	ip_proxy

With *Record-Route* disabled, the following IP rules must be added to those above:

	Action	Src Interface	Src Network	Dest Interface	Dest Network
OutboundBypassProxy	Allow	lan	lannet	wan	all-nets
InboundBypassProxy	Allow	wan	all-nets	lan	lannet

7.2.9. The H.323 ALG

H.323 is a standard approved by the International Telecommunication Union (ITU) 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). (For VoIP see also Section 7.2.8, “The SIP ALG”.)

H.323 Components

H.323 consists of four main components:

Terminals Devices used for audio and optionally video or data communication, such as phones, conferencing units, or "software phones" such as the product "NetMeeting".

Gateways 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 streams. A gateway is not

required for communication between two H.323 terminals.

Gatekeepers 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 signalling through itself to perform functions such as follow-me/find-me, forward on busy, etc. It is needed when there is more then one H.323 terminal behind a NATing device with only one public IP.

Multipoint Control Units 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:

H.225 RAS signalling and Call Control (Setup) signalling Used for call signalling. It is 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.

H.245 Media Control and Transport Provides control of multimedia sessions established between two H.323 endpoints. Its most important task is to negotiate opening and closing of logical channels. A logical channel could be, for example, an audio channel used for voice communication. Video and T.120 channels are also called logical channels during negotiation.

T.120 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 configuration of the standard H.323 ALG can be changed to suit different usage scenarios. The configurable options are:

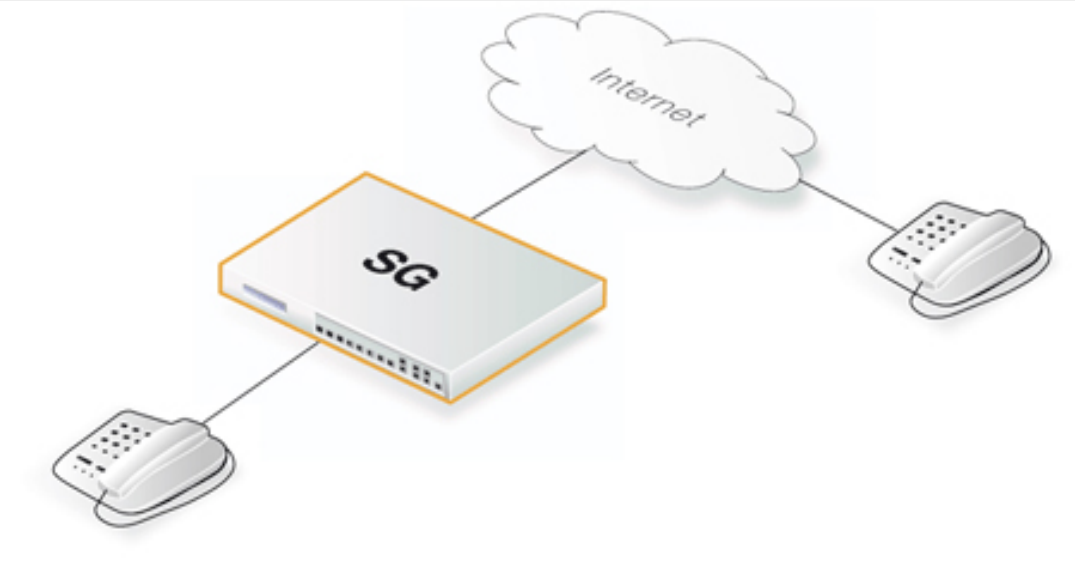
- **Allow TCP Data Channels** - This option allows TCP based data channels to be negotiated. Data channels are used, for example, by the T.120 protocol.
- **Number of TCP Data Channels** - The number of TCP data channels allowed can be specified.
- **Address Translation** - For NATed traffic the **Network** can be specified, which is what is allowed to be translated. The **External IP** for the **Network** is specified which is the IP address to NAT with. If the **External IP** is set as *Auto* then the external IP is found automatically through route lookup.
- **Translate Logical Channel Addresses** - This would normally always be set. If not enabled then no address translation will be done on logical channel addresses and the administrator needs to be sure about IP addresses and routes used in a particular scenario.
- **Gatekeeper Registration Lifetime** - The gatekeeper registration lifetime can be controlled in order to force re-registration by clients within a certain time. A shorter time forces more frequent registration by clients with the gatekeeper and less probability of a problem if the network becomes unavailable and the client thinks it is still registered.

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 7.6. 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.



Web Interface

Outgoing Rule:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323AllowOut
   - **Action:** Allow
   - **Service:** H323
   - **Source Interface:** lan
   - **Destination Interface:** any
   - **Source Network:** lannet
   - **Destination Network:** 0.0.0.0/0 (all-nets)
   - **Comment:** Allow outgoing calls
3. Click **OK**

Incoming Rule:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323AllowIn
   - **Action:** Allow
   - **Service:** H323
   - **Source Interface:** any
   - **Destination Interface:** lan
   - **Source Network:** 0.0.0.0/0 (all-nets)
   - **Destination Network:** lannet
   - **Comment:** Allow incoming calls
3. Click **OK**

Example 7.7. 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.

Web Interface

Outgoing Rule:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323Out
   - **Action:** NAT
   - **Service:** H323
   - **Source Interface:** lan
   - **Destination Interface:** any
   - **Source Network:** lannet
   - **Destination Network:** 0.0.0.0/0 (all-nets)
   - **Comment:** Allow outgoing calls
3. Click **OK**

Incoming Rules:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323In
   - **Action:** SAT
   - **Service:** H323
   - **Source Interface:** any
   - **Destination Interface:** core
   - **Source Network:** 0.0.0.0/0 (all-nets)
   - **Destination Network:** ip_wan (external IP of the security gateway)
   - **Comment:** Allow incoming calls to H.323 phone at ip-phone
3. For **SAT** enter **Translate Destination IP Address:** To New IP Address: ip-phone (IP address of phone)
4. Click **OK**

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323In
   - **Action:** Allow
   - **Service:** H323
   - **Source Interface:** any
   - **Destination Interface:** core

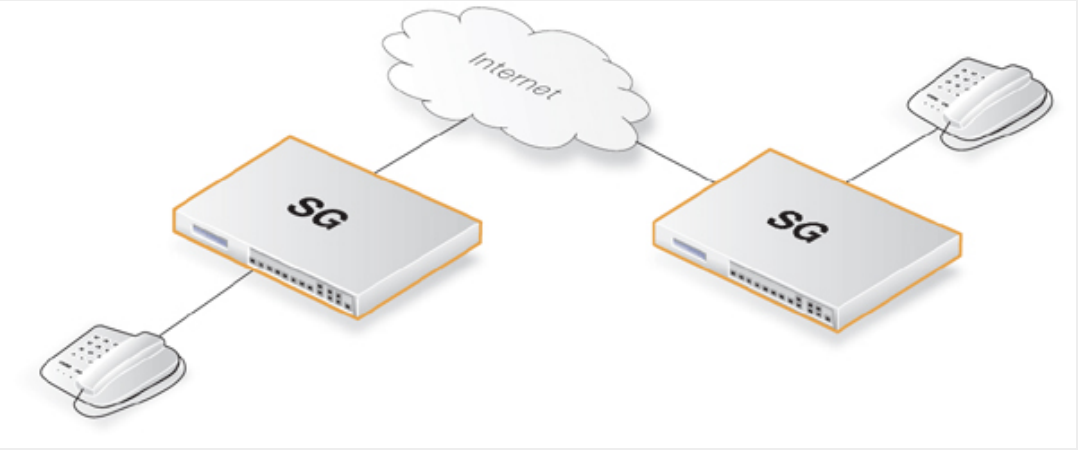
- **Source Network:** 0.0.0.0/0 (all-nets)
- **Destination Network:** ip_wan (external IP of the security gateway)
- **Comment:** Allow incoming calls to H.323 phone at ip-phone

3. Click **OK**

To place a call to the phone behind the Clavister Security Gateway, place a call to the external IP address on the security gateway. If multiple H.323 phones are placed behind the security 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 7.8. 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 security gateways. Make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules.



Web Interface

Outgoing Rule:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323AllowOut
   - **Action:** Allow
   - **Service:** H323
   - **Source Interface:** lan
   - **Destination Interface:** any
   - **Source Network:** lannet
   - **Destination Network:** 0.0.0.0/0 (all-nets)
   - **Comment:** Allow outgoing calls
3. Click **OK**

Incoming Rule:

1. Go to **Rules > IP Rules > Add > IPRule**

2. Now enter:
   - **Name:** H323AllowIn
   - **Action:** Allow
   - **Service:** H323
   - **Source Interface:** any
   - **Destination Interface:** lan
   - **Source Network:** 0.0.0.0/0 (all-nets)
   - **Destination Network:** lannet
   - **Comment:** Allow incoming calls
3. Click **OK**

Example 7.9. 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 security 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 security gateway.

Web Interface

Outgoing Rule:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323Out
   - **Action:** NAT
   - **Service:** H323
   - **Source Interface:** lan
   - **Destination Interface:** any
   - **Source Network:** lannet
   - **Destination Network:** 0.0.0.0/0 (all-nets)
   - **Comment:** Allow outgoing calls
3. Click **OK**

Incoming Rules:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323In
   - **Action:** SAT
   - **Service:** H323
   - **Source Interface:** any
   - **Destination Interface:** core
   - **Source Network:** 0.0.0.0/0 (all-nets)

- **Destination Network:** ip_wan (external IP of the security gateway)
- **Comment:** Allow incoming calls to H.323 phone at ip-phone

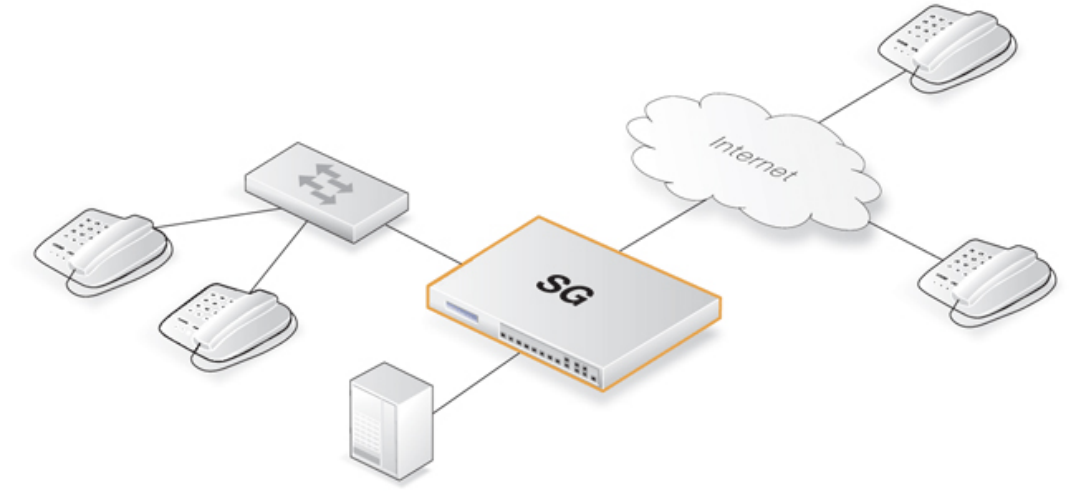
3. For **SAT** enter **Translate Destination IP Address:** To New IP Address: ip-phone (IP address of phone)
4. Click **OK**

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323In
   - **Action:** Allow
   - **Service:** H323
   - **Source Interface:** any
   - **Destination Interface:** core
   - **Source Network:** 0.0.0.0/0 (all-nets)
   - **Destination Network:** ip_wan (external IP of the security gateway)
   - **Comment:** Allow incoming calls to H.323 phone at ip-phone
3. Click **OK**

To place a call to the phone behind the Clavister Security Gateway, place a call to the external IP address on the security gateway. If multiple H.323 phones are placed behind the security 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 7.10. 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 security 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 security gateways, make sure there are no rules disallowing or allowing the same kind of ports/traffic before these rules.



Web Interface

Incoming Gatekeeper Rules:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323In
   - **Action:** SAT
   - **Service:** H323-Gatekeeper
   - **Source Interface:** any
   - **Destination Interface:** core
   - **Source Network:** 0.0.0.0/0 (all-nets)
   - **Destination Network:** ip_wan (external IP of the security gateway)
   - **Comment:** SAT rule for incoming communication with the Gatekeeper located at ip-gatekeeper
3. For **SAT** enter **Translate Destination IP Address:** To New IP Address: ip-gatekeeper (IP address of gatekeeper).
4. Click **OK**

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323In
   - **Action:** Allow
   - **Service:** H323-Gatekeeper
   - **Source Interface:** any
   - **Destination Interface:** core
   - **Source Network:** 0.0.0.0/0 (all-nets)
   - **Destination Network:** ip_wan (external IP of the security gateway)
   - **Comment:** Allow incoming communication with the Gatekeeper
3. Click **OK**

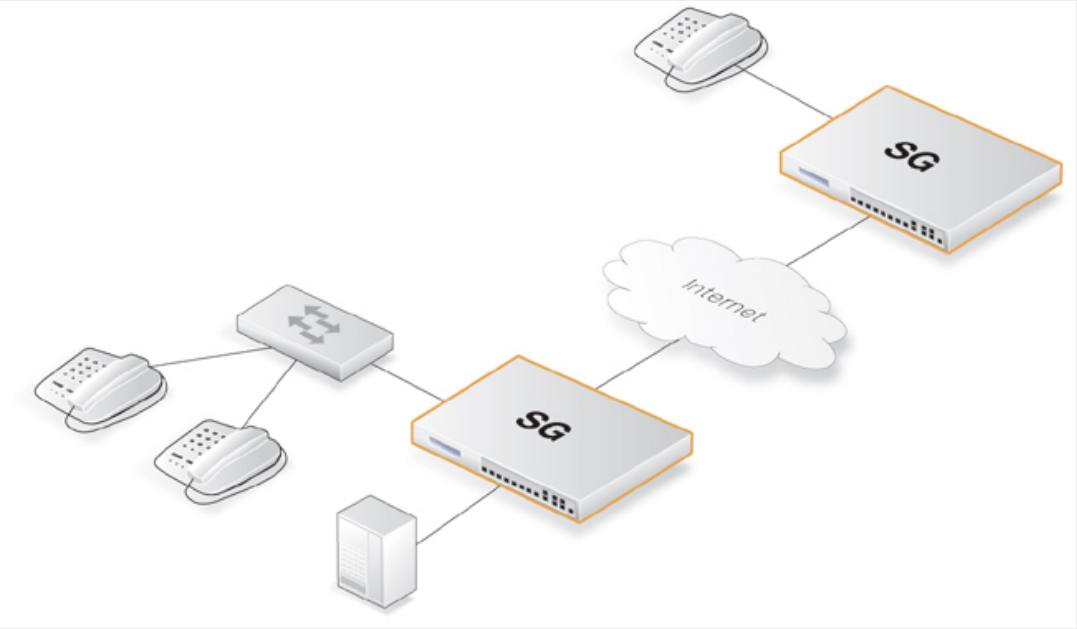
1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323In
   - **Action:** Allow
   - **Service:** H323-Gatekeeper
   - **Source Interface:** lan
   - **Destination Interface:** dmz
   - **Source Network:** lannet
   - **Destination Network:** ip-gatekeeper (IP address of the gatekeeper)
   - **Comment:** Allow incoming communication with the Gatekeeper
3. 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 is possible for internal phones to call the external phones that are registered with the gatekeeper.

Example 7.11. 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.



Web Interface

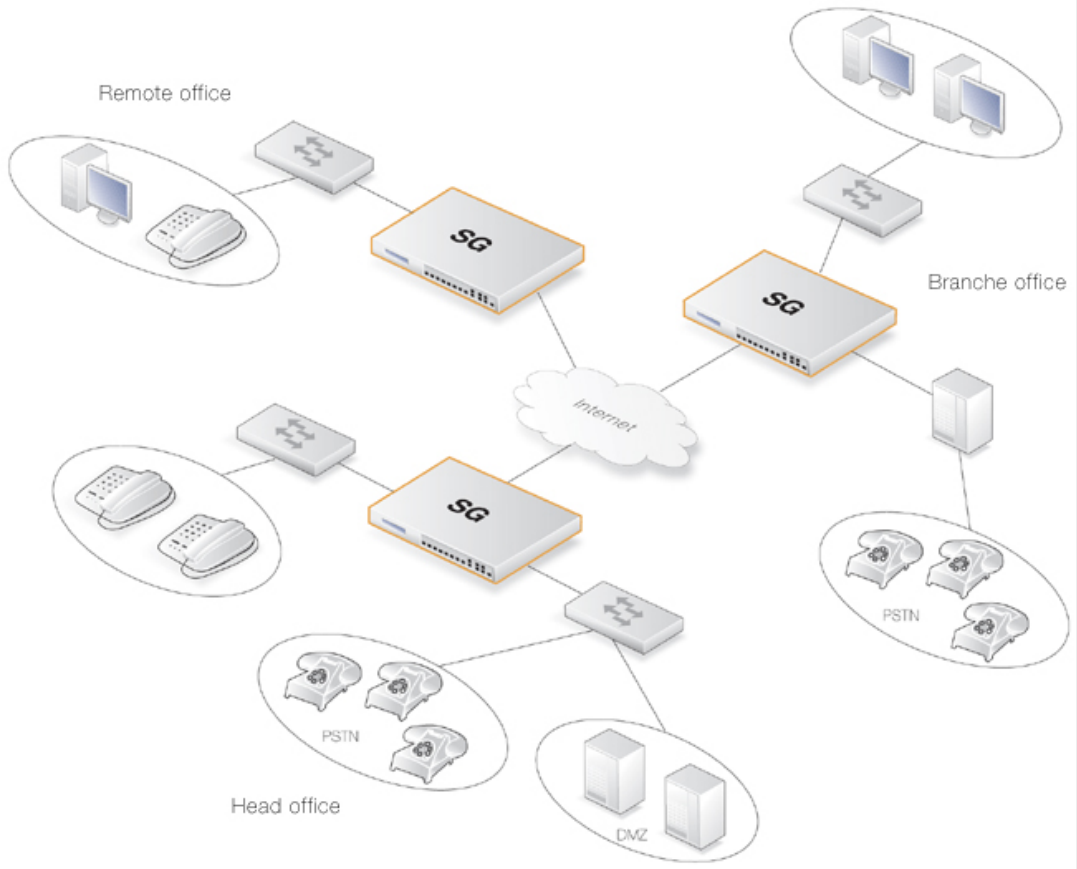
1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** H323Out
   - **Action:** NAT
   - **Service:** H323-Gatekeeper
   - **Source Interface:** lan
   - **Destination Interface:** any
   - **Source Network:** lannet
   - **Destination Network:** 0.0.0.0/0 (all-nets)
   - **Comment:** Allow outgoing communication with a gatekeeper
3. 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 is possible for internal phones to call the external phones that are registered with the gatekeeper.

Example 7.12. 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 security gateway should be configured as follows:

Web Interface

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** LanToGK
   - **Action:** Allow
   - **Service:** H323-Gatekeeper
   - **Source Interface:** lan
   - **Destination Interface:** dmz
   - **Source Network:** lannet
   - **Destination Network:** ip-gatekeeper
   - **Comment:** Allow H.323 entities on lannet to connect to the Gatekeeper
3. Click **OK**

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:

- **Name:** LanToGK
- **Action:** Allow
- **Service:** H323-Gatekeeper
- **Source Interface:** lan
- **Destination Interface:** dmz
- **Source Network:** lannet
- **Destination Network:** ip-gateway
- **Comment:** Allow H.323 entities on lannet to call phones connected to the H.323 Gateway on the DMZ

3. Click **OK**

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** GWToLan
   - **Action:** Allow
   - **Service:** H323-Gatekeeper
   - **Source Interface:** dmz
   - **Destination Interface:** lan
   - **Source Network:** ip-gateway
   - **Destination Network:** lannet
   - **Comment:** Allow communication from the Gateway to H.323 phones on lannet
3. Click **OK**

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** BranchToGW
   - **Action:** Allow
   - **Service:** H323-Gatekeeper
   - **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
3. Click **OK**

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** BranchToGW
   - **Action:** Allow
   - **Service:** H323-Gatekeeper
   - **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

3. Click **OK**

Example 7.13. 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 security gateways).

Web Interface

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** ToGK
   - **Action:** Allow
   - **Service:** H323-Gatekeeper
   - **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
3. Click **OK**

Example 7.14. 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:

Web Interface

1. Go to **Rules > IP Rules > Add > IPRule**
2. Now enter:
   - **Name:** GWToGK
   - **Action:** Allow
   - **Service:** H323-Gatekeeper
   - **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
3. 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 is possible for internal phones to call the external phones that are registered with the gatekeeper.

7.3. Web Content Filtering

7.3.1. 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.

Through the HTTP ALG, 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.

Note

All Web Content Filtering is enabled via the HTTP ALG which is described in Section 7.2.2, "The HTTP ALG".

7.3.2. 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 7.15. 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.

CLI

```
Device:/> set ALG ALG_HTTP content_filtering RemoveActiveX=Yes RemoveApplets=Yes
```

Web Interface

1. Go to **Objects > ALG**
2. In the table, click on our HTTP ALG object, *content_filtering*
3. Check the **Strip ActiveX objects (including flash)** control
4. Check the **Strip Java applets** control
5. Click **OK**

7.3.3. Static Content Filtering

Through the HTTP ALG, 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:

.example.com/	**Good.** This will block all hosts in the *example.com* domain and all web pages served by those hosts.
www.example.com/*	**Good.** This will block the *www.example.com* website and all web pages served by that site.
***/*.gif**	**Good.** This will block all files with *.gif* as the file name extension.
www.example.com	**Bad.** This will only block the first request to the web site. Surfing to *www.example.com/index.html*, for example, will not be blocked.
example.com/	**Bad.** This will also cause *www.myexample.com* to be blocked since it

blocks all sites ending with *example.com*.

Note: The hosts and networks blacklist is separate

Web content filtering URL blacklisting is a separate concept from Section 7.7, "Blacklisting Hosts and Networks".

Example 7.16. 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.

CLI

Start by adding an HTTP ALG in order to filter HTTP traffic:

```
Device:/> add ALG ALG_HTTP content_filtering
```

Then create a HTTP ALG URL to set up a blacklist:

```
Device:/> cc ALG ALG_HTTP content_filtering

Device:/content_filtering> add ALG_HTTP_URL URL=*/*.exe Action=Blacklist
```

Finally, make an exception from the blacklist by creating a specific whitelist:

```
Device:/content_filtering> add ALG_HTTP_URL URL=www.Clavister.com/*.exe
                                    Action=Whitelist
```

Web Interface

Start by adding an HTTP ALG in order to filter HTTP traffic:

1. Go to **Objects > ALG > Add > HTTP ALG**
2. Enter a suitable name for the ALG, for example *content_filtering*
3. Click **OK**

Then create a HTTP ALG URL to setup a blacklist:

1. Go to **Objects > ALG**
2. In the table, click on the recently created HTTP ALG to view its properties
3. Click the **HTTP URL** tab
4. Now click **Add** and select **HTTP ALG URL** from the menu
5. Select **Blacklist** as the **Action**
6. Enter **/*.exe* in the **URL** textbox
7. Click **OK**

Finally, make an exception from the blacklist by creating a whitelist:

1. Go to **Objects > ALG**
2. In the table, click on the recently created HTTP ALG to view its properties
3. Click the **HTTP URL** tab

4. Now click **Add** and select **HTTP ALG URL** from the menu
5. Select **Whitelist** as the **Action**
6. In the **URL** textbox, enter *www.Clavister.com/*.exe*
7. Click **OK**

Simply continue adding specific blacklists and whitelists until the filter satisfies the needs.

7.3.4. Dynamic Web Content Filtering

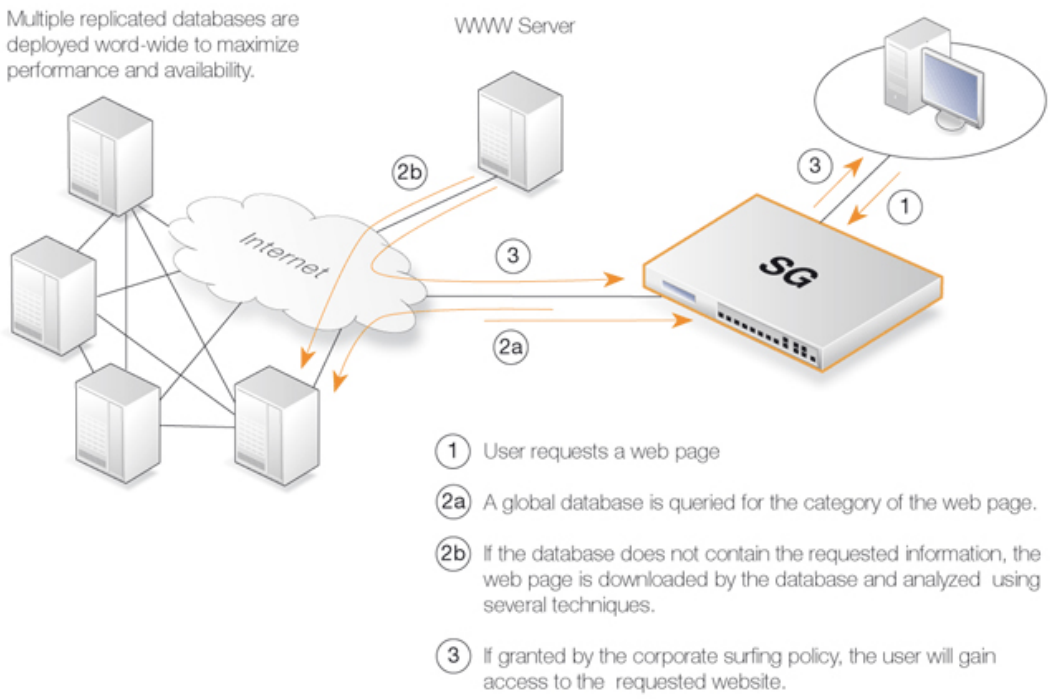
7.3.4.1. Overview

As part of the HTTP ALG, CorePlus supports *Dynamic Web Content Filtering* (WCF) 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 is 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 7.5. 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 URL submissions are done anonymously

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 are not blocked by the filtering policy.

WCF and Whitelisting

If a particular URL is whitelisted then it will bypass the WCF subsystem. No classification will be done on the URL and it will always be allowed. This applies if the URL has an exact match with an entry on the whitelist or if it matches an entry that makes use of wildcarding.

7.3.4.2. Setting Up Filtering

Activation

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, see Appendix A, *Subscribing to Security Updates*.

Once a subscription is taken out, an HTTP Application Layer Gateway (ALG) Object should be

defined with Dynamic Content Filtering enabled. This object is then associated with a Service object and the Service object is then associated with a rule in the IP rule set to determine which traffic should be subject to the filtering. This makes possible the setting up of a detailed 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 4.6, "Schedules".

Setting Fail Mode

The option exists to set the HTTP ALG *fail mode* in the same way that it can be set for some other ALGs and it applies to WCF just as it does to functions such as Anti-Virus scanning. The fail mode setting determines what happens when dynamic content filtering cannot function and, typically, this is because CorePlus is unable to reach the external databases to perform URL lookup. Fail mode can have one of two settings:

- **Deny** - If WCF is unable to function then URLs are denied if external database access to verify them is not possible. The user will see an "Access denied" web page.
- **Allow** - If the external WCF database is not accessible, URLs are allowed even though they might be disallowed if the WCF databases were accessible.

Example 7.17. Enabling Dynamic Web 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**.

CLI
(The NAT rule is called **NATHttp** for the CLI example)
First, create an HTTP Application Layer Gateway (ALG) Object:

```
Device:/> add ALG ALG_HTTP content_filtering WebContentFilteringMode=Enabled
            FilteringCategories=SEARCH_SITES
```

Then, create a Service object using the new HTTP ALG:

```
Device:/> add ServiceTCPUDP http_content_filtering Type=TCP DestinationPorts=80
            ALG=content_filtering
```

Finally, modify the NAT rule to use the new service:

```
Device:/> set IPRule NATHttp Service=http_content_filtering
```

Web Interface

First, create an HTTP Application Layer Gateway (ALG) Object:

1. Go to **Objects > ALG > Add > HTTP ALG**
2. Specify a suitable name for the ALG, for example *content_filtering*
3. Click the **Web Content Filtering** tab
4. Select **Enabled** in the **Mode** list
5. In the **Blocked Categories** list, select **Search Sites** and click the **>>** button.
6. Click **OK**

Then, create a Service object using the new HTTP ALG:

1. Go to **Local Objects > Services > Add > TCP/UDP service**
2. Specify a suitable name for the Service, for example *http_content_filtering*
3. Select the **TCP** in the **Type** dropdown list
4. Enter **80** in the **Destination Port** textbox
5. Select the HTTP ALG you just created in the **ALG** list
6. Click **OK**

Finally, modify the NAT rule to use the new service:

1. Go to **Rules > IP Rules**
2. In the grid control, click the NAT rule handling your HTTP traffic
3. Click the **Service** tab
4. Select your new service, http_content_filtering, in the pre-defined **Service** 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 example, *www.google.com*.
3. If everything is configured correctly, the web browser will present a web page that informs the user about that the requested site is blocked.

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 is then easier to have a good understanding of surfing behavior and also the potential time savings that can be made by enabling content filtering. It is 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 for better evaluation as to whether the goals of blocking are being met.

Example 7.18. Enabling Audit Mode

This example is based on the same scenario as the previous example, but now with audit mode enabled.

CLI
First, create an HTTP Application Layer Gateway (ALG) Object:

```
Device:/> add ALG ALG_HTTP content_filtering WebContentFilteringMode=Audit
            FilteringCategories=SEARCH_SITES
```

Web Interface

First, create an HTTP Application Layer Gateway (ALG) Object:

1. Go to **Objects > ALG > Add > HTTP ALG**

2. Specify a suitable name for the ALG, for example *content_filtering*
3. Click the **Web Content Filtering** tab
4. Select **Audit** in the **Mode** list
5. In the **Blocked Categories** list, select **Search Sites** and click the **>>** button
6. Click **OK**

The steps to then create a Service object using the new HTTP ALG and modifyng the NAT rule to use the new service, are described in the previous example.

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 will not 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

If a user overrides the restricted site notice page, they are allowed to surf to all pages without any new restricted site message appearing again. The user is however still being logged. When the user has become inactive for 5 minutes, the restricted site page will reappear if they then try to access a restricted site.

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 7.19. 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.

CLI
First, create an HTTP Application Layer Gateway (ALG) Object:

```
Device:/> add ALG ALG_HTTP content_filtering WebContentFilteringMode=Enable
            FilteringCategories=SEARCH_SITES AllowReclassification=Yes
```

Then, continue setting up the service object and modifying the NAT rule as we have done in the previous examples.

Web Interface

First, create an HTTP Application Layer Gateway (ALG) Object:

1. Go to **Objects > ALG > Add > HTTP ALG**
2. Specify a suitable name for the ALG, for example *content_filtering*
3. Click the **Web Content Filtering** tab
4. Select **Enabled** in the **Mode** list
5. In the **Blocked Categories** list, select **Search Sites** and click the **>>** button
6. Check the **Allow Reclassification** control
7. Click **OK**

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 example *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.

Event Messages

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.

7.3.4.3. 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
- 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 (for example, 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
- 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
- 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
- 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
- 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
- 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
- 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, online forums, discussion groups as well as URLs for downloading chat software. Examples might be:

- www.thetalkroom.org
- 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
- 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
- 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
- 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
- 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

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
- 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
- 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
- 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
- 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
- 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 organization. Examples might be:

- 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
- 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
- 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
- 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
- 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, for example 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
- 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 organizations. This category is populated by request or submission from various educational organizations. Examples might be:

- highschoolessays.org
- 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
- www.tripleclick.com

Category 28: Drugs/Alcohol

A web site may be classified under the Drugs/Alcohol category if its content includes drug and alcohol related information or services. Some URLs categorized under this category may also be categorized under the Health category. Examples might be:

- www.the-cocktail-guide.com
- 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
- www.gnu.org

Category 30: Swimsuit/Lingerie/Models

A web site may be categorized 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
- 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
- www.pleaseupdateyourdetails.com

Category 32: Non-Managed

Unclassified sites and sites that do not 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.

7.3.4.4. Customizing HTML Pages

Dynamic Web Content filtering make use of a set of HTML files to present information to the user when certain conditions occur such as trying to access a blocked site. These web pages, sometimes referred to as *HTTP banner files*, are stored within CorePlus but can be customized to suit a

particular installation's needs. The WebUI provides a simple way to download, edit and upload these files. The available files are:

CompressionForbidden
ContentForbidden
URLForbidden
RestrictedSiteNotice
ReclassifyURL

To perform customization it is necessary to first create a new, named **ALG Banner Files** object. This new object automatically contains a copy of all the files in the *Default* ALG Banner Files object. These new files can then be edited and uploaded back to CorePlus. The original *Default* object cannot be edited. The following example goes through the necessary steps.

Example 7.20. Editing Content Filtering HTTP Banner Files

This example shows how to modify the contents of the *URL forbidden* HTML page.

Web Interface

1. Go to **Objects > Banner files > Add > ALG Banner Files**
2. Enter a name such as *new_forbidden* and press **OK**
3. The dialog for the new set of ALG banner files will appear
4. Click the **Edit & Preview tab**
5. Select *URLForbidden* from the **Page** list
6. Now edit the HTML source that appears in the text box for the Forbidden URL page
7. Use **Preview** to check the layout if required
8. Press **Save** to save the changes
9. Click **OK** to exit editing
10. Go to **User Authentication > User Authentication Rules**
11. Select the relevant HTML ALG and click the **Agent Options** tab
12. Set the **HTTP Banners** option to be *new_forbidden*
13. Click **OK**
14. Go to **Configuration > Save & Activate** to activate the new file
15. Press **Save** and then click **OK**

The new file will be uploaded to CorePlus

Note
*In the above example, more than one HTML file can be edited in a session but the **Save** button should be pressed to save any edits before beginning editing on another file.*

Uploading with SCP

It is possible to upload new HTTP Banner files using SCP. The steps to do this are:

1. Since SCP cannot be used to download the original default HTML, the source code must be first copied from the WebUI and pasted into a local text file which is then edited using an appropriate editor.

2. A new **ALG Banner Files** object must exist which the edited file(s) is uploaded to. If the

object is called *mytxt*, the CLI command to create this object is:

```
Device:/> add HTTPALGBanners mytxt
```

This creates an object which contains a copy of all the *Default* content filtering banner files.

3. The modified file is then uploaded using SCP. It is uploaded to the object type *HTTPALGBanner* and the object *mytxt* with the property name *URLForbidden*. If the edited *URLForbidden* local file is called *my.html* then using the Open SSH SCP client, the upload command would be:

```
scp myhtml admin@10.5.62.11:HTTPAuthBanners/mytxt/URLForbidden
```

The usage of SCP clients is explained further in Section 3.1.6, “Secure Copy”.

4. Using the CLI, the relevant HTTP ALG should now be set to use the *mytxt* banner files. If the ALG us called *my_http_alg*, the command would be:

```
set ALG_HTTP my_http_alg HTTPBanners=mytxt
```

5. As usual, the *activate* followed by the *commit* CLI commands must be used to activate the changes on the Clavister Security Gateway.

HTML Page Parameters

The HTML pages contain a number of parameters that can be used as and where it is appropriate. The parameters available are:

- **%URL%** - The URL which was requested
- **%IPADDR%** - The IP address which is being browsed from
- **%REASON%** - The reason that access was denied

7.4. Anti-Virus Scanning

7.4.1. 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 focused on downloads by clients. CorePlus Anti-Virus is designed to be a complement to the standard antivirus scanning normally carried out locally by specialized 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 antivirus scanning is not available.

Enabling Through ALGs

CorePlus Anti-Virus is enabled on a per ALG basis. It is available for file downloads associated with the following ALGs and is enabled in the ALGs themselves:

- The HTTP ALG
- The FTP ALG
- The POP3 ALG
- The SMTP ALG

Hardware Acceleration

Anti-Virus performance, like IDP scanning performance, can be increased with an optional hardware acceleration upgrade on the SG50, SG4200 and SG4400 Security Gateways. Multiple streams are accelerated asynchronously in this optional hardware to boost overall throughput. The console command

```
Device:/> hwaccel
```

can be used to check if acceleration is installed. Note that Anti-Virus is not available on the discontinued SG30 model.

7.4.2. Implementation

Streaming

As a file transfer is streamed through the 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 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 the Clavister Security Gateway. Once a virus is recognized in the contents of a file, the download can be terminated before it completes.

Types of File Downloads Scanned

As described above, Anti-Virus scanning is enabled on a per ALG basis and can scan file downloads associated with the HTTP, FTP, SMTP and POP3 ALGs. More specifically:

- Any uncompressed file type transferred through these ALGs can be scanned.
- If the download has been compressed, ZIP and GZIP file downloads 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.

Protocol Specific behavior

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.

7.4.3. Activating Anti-Virus Scanning

Association with an ALG

Activation of Anti-Virus scanning is achieved through an ALG associated with the targeted protocol. An ALG object must first exist with the Anti-Virus option enabled. As always, an ALG must then be associated with an appropriate Service object for the protocol to be scanned. The Service object is then associated with a rule in the IP rule set which defines the origin and destination of the traffic to which the ALG is to be applied.

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 its 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.

7.4.4. 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.

7.4.5. 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*.

7.4.6. Anti-Virus Options

When configuring Anti-Virus scanning in an ALG, the following parameters can be set:

1. General options

Mode — This must be one of:
A. Disabled - Anti-Virus is switched off.
B. Audit - Scanning is active but logging is the only action.
C. Protect - Anti-Virus is active. Suspect files are dropped and logged.

Fail mode behavior — If a virus scan fails for any reason then the transfer can be dropped or allowed, with the event being logged. If this option is set to *Allow* then a condition such as the virus database not being available or the current license not being valid will not cause files to be dropped. Instead, they will be allowed through and a log message will be generated to indicate a failure has occured.

2. 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, such as image files in HTTP downloads.

CorePlus performs MIME content checking on all the filetypes listed in Appendix C, *Verified MIME filetypes* to establish the file's true filetype and then look for that filetype in the excluded list. If the file's type cannot be established from its contents (and this may happen with filetypes not specified in Appendix C, *Verified MIME filetypes*) then the filetype in the file's name is used when the excluded list is checked.

3. Compression Ratio Limit

When scanning compressed files, CorePlus must apply decompression to examine the file's contents. Some types of data can result in very high compression ratios where the compressed file is a small fraction of the original uncompressed file size. This can mean that a comparatively small compressed file attachment might need to be uncompressed into a much larger file which can place an excessive load on CorePlus resources and noticeably slowdown throughput.

To prevent this situation, the administrator should specify a *Compression Ratio* limit. If the limit of the ration is specified as **10** then this will mean that if the uncompressed file is 10 times larger than the compressed file, the specified Action should be taken. The Action can be one of:

- **Allow** - The file is allowed through without virus scanning
- **Scan** - Scan the file for viruses as normal
- **Drop** - Drop the file

In all three of the above cases the event is logged.

Verifying the MIME Type

The ALG **File Integrity** options can be utilized with Anti-Virus scanning to check that the file's contents matches the MIME type it claims to be.

The MIME type identifies a file's type. For instance a file might be identified as being of type *.gif* 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 *.gif* file but the file's data will not match that type's data pattern because it is infected with a virus.

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 C, *Verified MIME filetypes*.

Selecting the Anti-Virus Engine

If hardware based Anti-Virus acceleration is not available then the CorePlus Anti-Virus module offers a choice of Anti-Virus software scanning engines. The advanced setting **AVSW_Engine** allows selection of the engine best suited to the environment in which it is running. The options for this setting are:

- **Auto** - Select the engine automatically.
- **DFA** - Use the engine which maximizes scanning speed.
- **NFA** - Use the engine which minimizes memory usage.

These options have no meaning if Anti-Virus hardware acceleration is being used.

Setting the Correct System Time

It is important that a CorePlus has the correct system time set if the auto-update feature in the Anti-Virus module can function correctly. An incorrect time can mean the auto-updating is disabled.

The console command

```
> updatecenter -status
```

will show the current status of the auto-update feature.

Updating in High Availability Clusters

Updating the Anti-Virus databases for both the Clavister Security Gateways in an HA Cluster is performed automatically by CorePlus. In a cluster there is always an *active* unit and an *inactive* unit. Only the active unit in the cluster will perform regular checking for new database updates. If a new database update becomes available the sequence of events will be as follows:

1. The active unit determines there is a new update and downloads the required files for the update.
2. The active unit performs an automatic reconfiguration to update its database.
3. This reconfiguration causes a failover so the passive unit becomes the active unit.
4. When the update is completed, the newly active unit also downloads the files for the update and performs a reconfiguration.
5. This second reconfiguration causes another failover so the passive unit reverts back to being active again.

These steps result in both Clavister Security Gateways in a cluster having updated databases and with the original active/passive roles. For more information about HA clusters refer to Chapter 12, *High Availability*.

Example 7.21. Activating 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 NAT this traffic.

CLI
First, create an HTTP Application Layer Gateway (ALG) Object with Anti-Virus scanning enabled:

```
Device:/> set ALG ALG_HTTP anti_virus Antivirus=Protect
```

Then, create a Service object using the new HTTP ALG:

```
Device:/> add ServiceTCPUDP http_anti_virus Type=TCP DestinationPorts=80
     ALG=anti_virus
```

Finally, modify the NAT rule to use the new service:

```
Device:/> set IPRule NATHttp Service=http_anti_virus
```

Web Interface

A. First, create an HTTP ALG Object:

1. Go to **Objects > ALG > Add > HTTP ALG**
2. Specify a suitable name for the ALG, for instance *anti_virus*
3. Click the **Antivirus** tab
4. Select **Protect** in the **Mode** dropdown list
5. Click **OK**

B. Then, create a Service object using the new HTTP ALG:

1. Go to **Local Objects > Services > Add > TCP/UDP service**
2. Specify a suitable name for the Service, for instance http_anti_virus
3. Select the **TCP** in the **Type** dropdown list
4. Enter **80** in the **Destination Port** textbox
5. Select the HTTP ALG you just created in the **ALG** dropdown list

6. Click **OK**

C. Finally, modify the NAT rule (called **NATHttp** in this example) to use the new service:

1. Go to **Rules > IP Rules**
2. In the grid control, click the NAT rule handling the traffic between **lannet** and **all-nets**
3. Click the **Service** tab
4. Select your new service, http_anti_virus, in the pre-defined **Service** dropdown list
5. Click **OK**

Anti-Virus scanning is now activated for all web traffic from **lannet** to **all-nets**.

7.5. Intrusion Detection and Prevention

7.5.1. 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 system. 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.

Note

IDP performance, like Anti-Virus scanning performance, can be increased with an optional hardware acceleration upgrade on the SG50, SG4200 and SG4400 Security Gateways. Multiple streams are accelerated asynchronously in this optional hardware to boost overall throughput. The console command ***hwaccel*** *can be used to check if acceleration is installed. Also note that IDP is not available on the discontinued SG30 model.*

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 search for in that traffic?
3. What action should be carried out when an intrusion is detected?

CorePlus IDP Components

CorePlus IDP addresses the above issues with the following mechanisms:

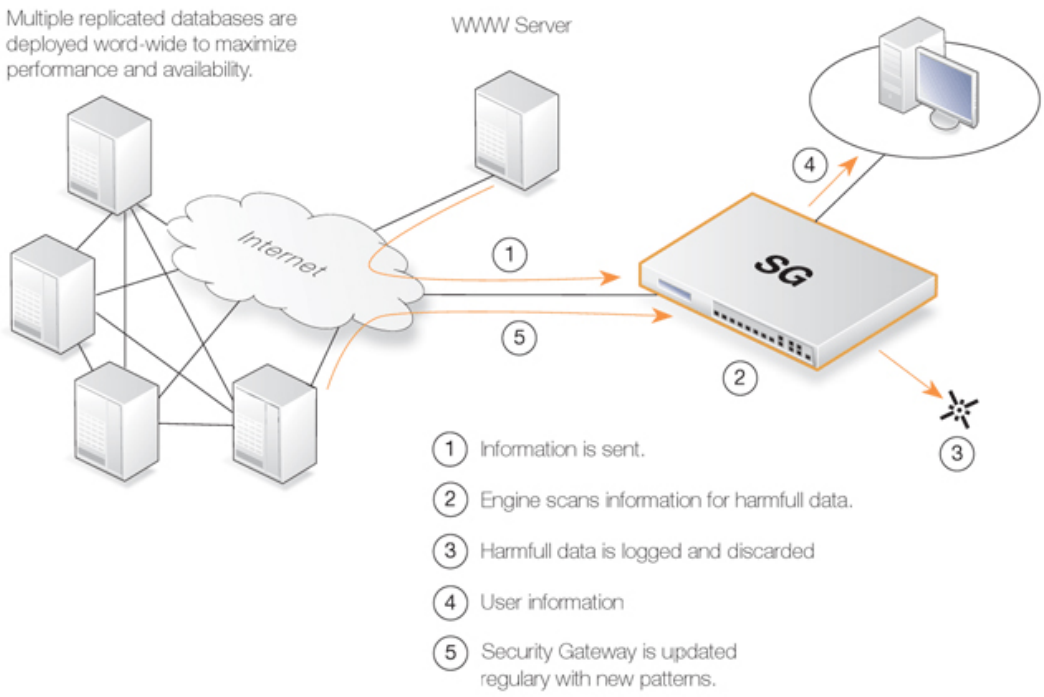
1. **IDP Rules** are defined up by the administrator to determine what traffic should be scanned.
2. **Pattern Matching** is applied by CorePlus IDP to the traffic that matches an IDP Rule as it streams through the gateway.
3. If CorePlus IDP detects an intrusion then the **Action** specified for the triggering IDP Rule is taken.

IDP Rules, Pattern Matching and IDP Rule Actions are described in the sections which follow.

7.5.2. 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 7.6. IDP Database Updating



Automatic 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. Auto-update is an option that can be enabled or disabled by the administrator.

With auto-update enabled, signature database updates are downloaded automatically by CorePlus 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 new signatures then new updates will be automatically downloaded, replacing any older versions. No reconfiguration is needed by the administrator to activate new signatures.

If auto-update is disabled then updates must be explicitly forced and the administrator needs to be aware of when new updates are available. However, this approach does help with more quickly detecting any false positives that new signatures might produce.

Setting the Correct System Time

It is important that a CorePlus has the correct system time set if the auto-update feature in the IDP

module can function correctly. An incorrect time can mean the auto-updating is disabled.

The console command

```
> updatecenter -status
```

will show the current status of the auto-update feature.

Updating in High Availability Clusters

Updating the IDP databases for both the Clavister Security Gateways in an HA Cluster is performed automatically by CorePlus. In a cluster there is always an *active* unit and an *inactive* unit. Only the active unit in the cluster will perform regular checking for new database updates. If a new database update becomes available the sequence of events will be as follows:

1. The active unit determines there is a new update and downloads the required files for the update.
2. The active unit performs an automatic reconfiguration to update its database.
3. This reconfiguration causes a failover so the passive unit becomes the active unit.
4. When the update is completed, the newly active unit also downloads the files for the update and performs a reconfiguration.
5. This second reconfiguration causes another failover so the passive unit reverts back to being active again.

These steps result in both Clavister Security Gateways in a cluster having updated databases and with the original active/passive roles. For more information about HA clusters refer to Chapter 12, *High Availability*.

7.5.3. 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. IDP Rules are constructed like other security policies in CorePlus such as IP Rules. 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.

HTTP Normalization

Each IDP rule has a section of settings for *HTTP normalization*. This allows the administrator to choose the actions that should be taken when IDP finds inconsistencies in the URIs embedded in incoming HTTP requests. Some server attacks are based on creating URIs with sequences that can exploit weaknesses in some HTTP server products.

The URI conditions which IDP can detect are:

- **Invalid UTF8**

  This looks for any invalid UTF8 characters in a URI.

- **Invalid hex encoding**

  A valid hex sequence is where a percentage sign is followed by two hexadecimal values to

represent a single byte of data. An invalid hex sequence would be percentage sign followed by something which is not a valid hexadecimal value.

- **Double encoding**

  This looks for any hex sequence which itself is encoded using other hex escape sequences. An example would be the original sequence *%2526* where *%25* is then might be decoded by the HTTP server to '*%*' and results in the sequence '*%26*'. This is then finally decoded to '*&*'.

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 such as address translation and logging.

7.5.4. Insertion/Evasion Attack Prevention

Overview

When defining an IDP Rule, the administrator has the option to enable or disable the ability to "Protect against Insertion/Evasion attack". *Insertion/Evasion Attack* is a form of attack which is specifically aimed at IDP systems. It exploits the fact that in a TCP/IP data transfer, the data stream must often be reassembled from smaller pieces of data because the individual pieces either arrive in the wrong order or are fragmented in some way. *Insertions* or *Evasions* are designed to exploit this reassembly process.

Insertion Attacks

An Insertion attack consists of inserting data into a stream so that the resulting sequence of data packets is accepted by the IDP subsystem but will be rejected by the targeted application. This results is two different streams of data.

As an example, consider a data stream broken up into 4 packets: p1, p2, p3 and p4. The attacker might first send packets p1 and p4 to the targeted application. These will be held by both the IDP subsystem and the application until packets p2 and p3 arrive so that reassembly can be done. The attacker now deliberately sends two packets, p2' and p3', which will be rejected by the application but accepted by the IDP system. The IDP system is now able to complete reassembly of the packets and believes it has the full data stream. The attacker now sends two further packets, p2 and p3, which will be accepted by the application which can now complete reassembly but resulting in a different data stream to that seen by the IDP subsystem.

Evasion Attacks

An evasion attack has a similar end-result to the Insertion Attack in that it also generates two different data streams, one that the IDP subsystem sees and one that the target application sees, but

it is achieved in the reverse way. It consists of sending data packets that are rejected by the IDP subsystem but are acceptable to the target application.

Detection Action

If an Insertion/Evasion Attack is detected with the Insertion/Evasion Protect option enabled, CorePlus automatically corrects the data stream by removing the extraneous data associated with the attack.

Insertion/Evasion Log Events

The Insertion/Evasion Attack subsystem in CorePlus can generate two types of log message:

- An **Attack Detected** log message, indicating an attack has been identified and prevented.
- An **Unable to Detect** log message when CorePlus has been unable to identify potential attacks when reassembling a TCP/IP stream although such an attack may have been present. This condition is caused by infrequent and unusually complex patterns of data in the stream.

Recommended Configuration

By default, Insertion/Evasion protection is enabled for all IDP rules and this is the recommended setting for most configurations. There are two motivations for disabling the option:

- **Increasing throughput** - Where the highest throughout possible is desirable, then turning the option off, can provide a slight increase in processing speed.
- **Excessive False Positives** - If there is evidence of an unusually high level of Insertion/Evasion false positives then disabling the option may be prudent while the false positive causes are investigated.

7.5.5. 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.

Recognizing 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 three signature types which offer differing levels of certainty with regard to threats:

- **Intrusion Protection Signatures (IPS)** - These 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)** - These 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** - These detect different types of application traffic. They can be used to block certain applications such as file sharing applications and instant messaging.

7.5.6. 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.

Specifying Signature Groups

IDP Signature Groups fall into a three level hierarchical structure. The top level of this hierarchy is the signature *Type*, the second level the *Category* and the third level the *Sub-Category*. The signature group called **POLICY_DB_MSSQL** illustrates this principle where **Policy** is the *Type*, **DB** is the *Category* and **MSSQL** is the *Sub-Category*. These 3 signature components are explained below:

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, for example *MSSQL*. The Sub-Category may not be necessary if the *Type* and *Category* are sufficient to specify the group, for example **APP_ITUNES**.

Listing of IDP Groups

A listing of IDP groupings can be found in Appendix B, *IDP Signature Groups*. The listing shows group 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 is 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.

IDP Signature Wildcarding

When selecting IDP signature groups, it is possible to use wildcarding to select more than one group. The "**?**" character can be used to wildcard for a single character in a group name. Alternatively, the "*****" character can be used to wildcard for any set of characters of any length in a group name.

Caution: Use the minimum IDP signatures necessary

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. For instance, using IDS_WEB, IPS_WEB*, IDS_HTTP* and IPS_HTTP* IDP groups would be appropriate for protecting an HTTP server.*

IDP traffic scanning creates an additional load on the hardware that in most cases should not noticeably degrade performance. Using too many signatures during scanning can make the load on the gateway hardware unnecessarily high, adversely affecting throughput.

7.5.7. IDP Actions

Action Options

After pattern matching recognizes 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

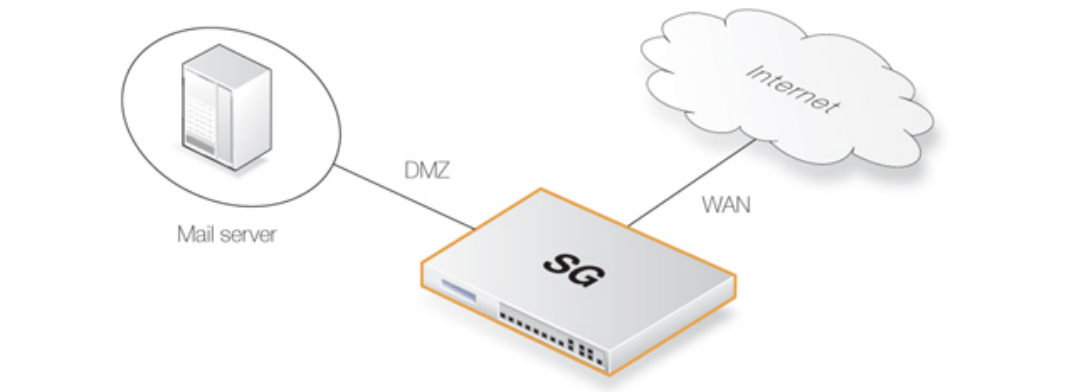
The **Protect** option includes the option that the particular host or network that triggers the IDP Rule can be added to a *Blacklist* of offending traffic sources. This means that all subsequent traffic coming from a blacklisted source with be automatically dropped by CorePlus. For more details of how blacklisting functions see Section 7.7, “Blacklisting Hosts and Networks”.

Tip

Any IP address that exists in the CorePlus whitelist cannot be blacklisted. For this reason it is recommended that the IP address of the management workstation and the Clavister Security Gateway itself is added to the whitelist when using IDP.

Example 7.22. 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.



CLI

Create IDP Rule:

```
Device:/> add IDPRule Service=smtp SourceInterface=wan SourceNetwork=wannet
     DestinationInterface=dmz DestinationNetwork=ip_mailserver
     Name=IDPMailSrvRule
```

Create IDP Action:

```
Device:/> cc IDPRule IDPMailSrvRule


Device:/IDPMailSrvRule> add IDPRuleAction Action=Protect
     IDPServity=All Signatures=IPS_MAIL_SMTP
```

Web Interface

Create IDP Rule:

This IDP rule will be called **IDPMailSrvRule**, and applies to the SMTP service. Source Interface and Source Network define 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 **IDP > IDP Rules > Add > IDP Rule**
2. Now enter:
   - **Name:** IDPMailSrvRule
   - **Service:** smtp
   - Also inspect dropped packets: In case all traffic matching this rule should be scanned (this also means traffic that the main rule set would drop), the "Also inspect dropped packets" checkbox should be checked, which is the case in this example.

- **Source Interface:** wan
- **Source Network:** wannet
- **Destination Interface:** dmz
- **Destination Network:** ip_mailserver
- Click **OK**

If logging of intrusion attempts is desired, this can be configured in the **Log Settings** tab.

Create IDP Action:

When this IDP Rule has been created, an action must also be created, specifying what signatures the IDP should use when scanning data matching the IDP Rule, and what CorePlus should do in case an intrusion is discovered. Intrusion attempts should cause the connection to be dropped, so **Action** is set to **Protect**. **Severity** is set to **Attack**, in order to match all SMTP attacks. **Signatures** is set to **IPS_MAIL_SMTP** in order to use signatures that describe attacks from the external network, dealing with the SMTP protocol.

1. Go to **IDP > IDP Rules > IDPMailSrvRule > Add > IDP Rule Action**
2. Now enter:
   - **Action:** Protect
   - **Severity:** All
   - **Signatures:** IPS_MAIL_SMTP
   - Click **OK**

In summary, the following will occur: If traffic from the external network to the mail server occurs, IDP will be activated. If traffic matches any of the signatures in the **IPS_MAIL_SMTP** signature group, the connection will be dropped, thus protecting the mail server.

7.6. Denial-Of-Service (DoS) Attacks

7.6.1. Overview

By embracing the Internet, enterprises experience new business opportunities and growth. The enterprise network and the applications that run over it are business critical. 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 use the same public infrastructure to mount attacks. Attack tools are readily available on the Internet and development work on these tools is often split across groups of novice hackers — known as "script kiddies" or "larval hackers" — scattered across the globe, providing around-the-clock progression of automated attack methods. Many of the new attack methods utilize the distributed nature of the Internet to launch DoS attacks against organizations.

To be on the receiving end of a DoS attack is probably the last thing any network administrator wants to experience. Attacks can appear out of thin air and the consequences can be devastating with crashed servers, jammed Internet connections and business critical systems in overload.

This section deals with using the Clavister Security Gateway to protect organizations against DoS attacks.

7.6.2. DoS Attack Mechanisms

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.

Some of the most commonly used DoS attacks have been:

- The Ping of Death / Jolt attacks
- Fragmentation overlap attacks: Teardrop / Bonk / Boink / Nestea
- The Land and LaTierra attacks
- The WinNuke attack
- Amplification attacks: Smurf, Papasmurf, Fraggle
- TCP SYN Flood attack
- The Jolt2 attack

7.6.3. *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 Advanced Settings.

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.

7.6.4. Fragmentation overlap attacks: *Teardrop, Bonk, Boink and Nestea*

Teardrop and its followers are fragment overlap attacks. 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 "IllegalFrags". The sender IP address may be spoofed.

7.6.5. 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. 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 here since it is always the same as the destination IP address.

7.6.6. 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.

7.6.7. 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 on the Victim's Side

Smurf, and its followers, are resource exhaustion attacks in that they use up Internet connection capacity. 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 targeted by the attacker. Tightening the inbound rule set may help.

The Traffic Shaping feature built into CorePlus also help absorb some of the flood before it reaches protected servers.

7.6.8. TCP SYN Flood Attacks

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 it is enabled in a Service object associated with the rule in the IP rule set that allows the traffic. By default, this is the case for the pre-defined services **http-in**, **https-in**, **smtp-in**, and **ssh-in**. If a new custom Service object is defined by the administrator then Syn Flood Protection can be enabled or disabled as desired.

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.

It should be noted that if Syn Flood Protection is enabled on a Service object and that Service object has an ALG associated with it then the ALG will be disabled.

7.6.9. 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 queued, 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.

7.6.10. Distributed DoS 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 installs 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 often prefer university or institutional networks because of their open, distributed nature. Tools used to launch DDoS attacks include Trin00, TribeFlood Network (TFN), TFN2K and Stacheldraht.

7.7. Blacklisting Hosts and Networks

Overview

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 subsystems have the ability to optionally blacklist a host or network when certain conditions are encountered. These subsystems are:

- Intrusion Detection and Prevention (IDP).
- Threshold Rules.

Blacklisting Options

The automatic blacklisting of a host or network can be enabled in IDP and in Threshold Rules by specifying the **Protect** action for when a rule is triggered. Once enabled there are three blacklisting options:

Time to Block Host/Network in seconds	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 its original, full value (in other words, it is not cumulative).
Block only this Service	By default Blacklisting blocks all Services for the triggering host.
Exempt already established connections from Blacklisting	If there are established connections that have the same source as this new Blacklist entry then they will not be dropped if this option is set.

IP addresses or networks are added to the list then the traffic from these sources is then blocked for the period of time specified.

Note: Restarts do not effect the blacklist

The contents of the blacklist is not lost if the Clavister Security Gateway shuts down and restarts.

Whitelisting

To ensure that Internet traffic coming from trusted sources, such as the management workstation, are not blacklisted under any circumstances, a *Whitelist* is also maintained by CorePlus. Any IP address object can be added to this whitelist

Tip: Important IP addresses should be whitelisted

It is recommended to add the Clavister Security Gateway itself to the whitelist as well as the IP address or network of the management workstation since blacklisting of either could have serious consequences for network operations.

It is also important to understand that although whitelisting prevents a particular source from being blacklisted, it still does not prevent CorePlus mechanisms such as Threshold Rules from dropping or denying connections from that source. What whitelisting does is prevent a source being added to a blacklist if that is the action a rule has specified.

For further details on usage see Section 7.5.7, “IDP Actions”, Section 11.2.7, “Threshold Rule Blacklisting” and Section 11.2, “Threshold Rules”.

Note: The content filtering blacklist is separate

Content filtering blacklisting is a separate subject and uses a separate logical list (see Section 7.3, “Web Content Filtering”).

The CLI *blacklist* Command

The *blacklist* command can be used to look at as well as manipulate the current contents of the blacklist and the whitelist. The current blacklist can be viewed with the command:

```
Device:/> blacklist -show -black
```

This *blacklist* command can be used to remove a host from the blacklist using the *-unblock* option.

Example 7.23. Adding a Host to the Whitelist

In this example we will add an IP address object called *white_ip* to the whitelist. This will mean this IP address can never be blacklisted.

CLI

```
Device:/> add BlacklistWhiteHost Addresses=white_ip Service=all_tcp
```

Web Interface

1. Goto **System > Whitelist > Add > Whitelist host**
2. Now select the IP address object *white_ip* so it is added to the whitelist
3. Select the service *all_tcp* to be associated with this whitelist entry
4. Click **OK**

Chapter 8. Address Translation

This chapter describes CorePlus address translation capabilities.

The ability of CorePlus to change the IP address of packets as they pass through the Clavister Security Gateway is known as *address translation*. CorePlus supports two types of translation: *Dynamic Network Address Translation* (NAT) and *Static Address Translation* (SAT). Both translations are policy-based meaning that they can be applied to specific traffic based on source/destination network/interface as well as service. Two types of IP rules, *NAT rules* and *SAT rules*, are used to specify address translation within the IP rule set.

There are two main reasons for employing address translation:

- **Functionality**

  Perhaps private IP addresses are used on a protected network and protected hosts need 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 the solution.

- **Security**

  Address translation does not, in itself provide any greater level of security, but it can make it more difficult for intruders to understand the exact layout of the protected network and which machines are susceptible to attack. In the worst case scenario, employing address translation will mean that an attack will take longer, which will also make it more visible in CorePlus log files. In the best-case scenario, an 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 configuring NAT and SAT rules.

8.1. Dynamic Network Address Translation

Dynamic Network Address Translation (NAT) provides a mechanism for translating original source IP addresses to a different addresses. The most common usage for NAT is when using private IP addresses in an internal network and it is desirable that outbound connections appear as though they originate from the Clavister Security Gateway itself instead of the internal addresses.

Many-To-One Translation

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 and usually this port is one equal to or above port number 32,768 (in other words the upper half of the total 65,536 port number range). This means that there is a limitation of a maximum of 32,768 simultaneous NATed connections that can use the same translated source IP address. This normally is normally adequate for all but the most extreme usage scenarios.

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, for example 192.168.1.5, sends a packet from a dynamically assigned port, for instance, port 1038, to a server, for example 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 8.1. Adding a NAT Rule

To add a NAT rule that will perform address translation for all HTTP traffic originating from the internal network, follow the steps outlined below:

CLI

First, change the current category to be the *main* IP rule set:

```
Device:/> cc IPRuleSet main
```

Now, create the IP rule:

```
Device:/main> add IPRule Action=NAT Service=http SourceInterface=lan
     SourceNetwork=lannet DestinationInterface=any
     DestinationNetwork=all-nets Name=NAT_HTTP NATAction=UseInterfaceAddress
```

Return to the top level:

```
Device:/main> cc
```

Web Interface

1. Go to **Rules > IP Rules > Add > IPRule**

2. Specify a suitable name for the rule, for example *NAT_HTTP*
3. Now enter:
   - **Action:** NAT
   - **Service:** http
   - **Source Interface:** lan
   - **Source Network:** lannet
   - **Destination Interface:** any
   - **Destination Network:** all-nets
4. Under the **NAT** tab, make sure that the **Use Interface Address** option is selected
5. Click **OK**

Protocols Handled by NAT

Dynamic address translation is able to deal with the TCP, UDP and ICMP protocols with a good level of functionality since the algorithm knows which values 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 servers 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, such as OSPF, L2TP, etc. They do not apply to the protocols transported by TCP, UDP and ICMP such as telnet, FTP, HTTP and SMTP. 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.

8.2. NAT Pools

Overview

As discussed in Section 8.1, “Dynamic Network Address Translation”, NAT provides a way to have multiple internal clients and hosts with unique private internal IP addresses communicate to remote hosts through a single external public IP address. When multiple public external IP addresses are available then a *NAT Pool* object can be used to allocate new connections across these public IP addresses.

NAT Pools are usually employed when there is a requirement for huge numbers of unique port connections. The CorePlus Port Manager has a limit of approximately 65,000 connections for a unique combination of source and destination IP addresses. Where large number of internal clients are using applications such as file sharing software, very large numbers of ports can be required for each client. The situation can be similarly demanding if a large number of clients are accessing the Internet through a proxy-server. The port number limitation is overcome by allocating extra external IP addresses for Internet access and using NAT Pools to allocate new connections across them.

Types of NAT Pools

A NAT Pool can be one of three types, each allocating new connections in a different way:

- **Stateful**
- **Stateless**
- **Fixed**

These three types are discussed below.

Stateful NAT Pools

When the *Stateful* option is selected, CorePlus allocates a new connection to the external IP address that currently has the least number of connections routed through it with the assumption that it is the least loaded. CorePlus keeps a record in memory of all such connections. Subsequent connections involving the same internal client/host will then use the same external IP address.

The advantage of the stateful approach is that it can balance connections across several external ISP links while ensuring that an external host will always communicate back to the same IP address which will be essential with protocols such as HTTP when cookies are involved. The disadvantage is the extra memory required by CorePlus to track the usage in its state table and the small processing overhead involved in processing a new connection.

To make sure that the state table does not contain dead entries for communications that are no longer active, a **State Keepalive** time can be specified. This time is the number of seconds of inactivity that must occur before a state in the state table is removed. After this period CorePlus assumes no more communication will originate from the associated internal host. Once the state is removed then subsequent communication from the host will result in a new state table entry and may be allocated to a different external IP address in the NAT Pool.

The state table itself takes up memory so it is possible to limit its size using the *Max States* value in a NAT Pool object. The state table is not allocated all at once but is incremented in size as needed. One entry in the state table tracks all the connections for a single host behind the Clavister Security Gateway no matter which external host the connection concerns. If *Max States* is reached then an existing state with the longest idle time is replaced. If all states in the table is active then the new connection is dropped. As a rule of thumb, the *Max States* value should be at least the number of local hosts or clients that will connect to the Internet.

There is only one state table per NAT Pool so that if a single NAT Pool is re-used in multiple NAT

IP rules they share the same state table.

Stateless NAT Pools

The *Stateless* option means that no state table is maintained and the external IP address chosen for each new connection is the one that has the least connections already allocated to it. This means two connections between one internal host to the same external host may use two different external IP addresses.

The advantage of a Stateless NAT Pool is that there is good spreading of new connections between external IP addresses with no requirement for memory allocated to a state table and there is less processing time involved in setting up each new connection. The disadvantage is that it is not suitable for communication that requires a constant external IP address.

Fixed NAT Pools

The *Fixed* option means that each internal client or host is allocated one of the external IP addresses through a hashing algorithm. Although the administrator has no control over which of the external connections will be used, this scheme ensures that a particular internal client or host will always communicate through the same external IP address.

The Fixed option has the advantage of not requiring memory for a state table and providing very fast processing for new connection establishment. Although explicit load balancing is not part of this option, there should be spreading of the load across the external connections due to the random nature of the allocating algorithm.

IP Pool Usage

When allocating external IP addresses to a NAT Pool it is not necessary to explicitly state these. Instead a CorePlus *IP Pool* object can be selected. IP Pools gather collections of IP addresses automatically through DHCP and can therefore supply external IP addresses automatically to a NAT Pool. See Section 6.5, “IP Pools” for more details on this topic.

Proxy ARP Usage

Where an external router sends ARP queries to the Clavister Security Gateway to resolve external IP addresses included in a NAT Pool, CorePlus will need to send the correct ARP replies for this resolution to take place through its Proxy ARP mechanism so the external router can correctly build its routing table.

By default, the administrator must specify in NAT Pool setup which interfaces will be used by NAT pools. The option exists however to enable Proxy ARP for a NAT Pool on all interfaces but this can cause problems sometimes by possibly creating routes to interfaces on which packets should not arrive. It is therefore recommended that the interface(s) to be used for the NAT Pool Proxy ARP mechanism are explicitly specified.

Using NAT Pools

NAT Pools are used in conjunction with a normal NAT IP rule. When defining a NAT rule, the dialog includes the option to select a NAT Pool to use with the rule. This association brings the NAT Pool into use.

Example 8.2. Using NAT Pools

This example creates a NAT pool which will be applied the external IP address range 10.6.13.10 to 10.16.13.15

and then uses it in a NAT IP rule for HTTP traffic on the **Wan** interface.

Web Interface

A. First create an object in the address book for the address range:

1. Go to **Objects > Address Book > Add > IP address**
2. Specify a suitable name for the IP range *nat_pool_range*
3. Enter *10.6.13.10-10.16.13.15* in the **IP Address** textbox
   (a network eg 10.6.13.0/24 could be used here - the 0 and 255 addresses will be automatically removed)
4. Click **OK**

B. Next create a Stateful NAT Pool object called *stateful_natpool* :

1. Go to **Objects > NAT Pools > Add > NAT Pool**
2. Now enter:
   - **Name:** stateful_natpool
   - **Pool type:** stateful
   - **IP Range:** nat_pool_range
3. Select the **Proxy ARP** tab and add the **WAN** interface
4. Click **OK**

C. Now define the NAT rule in the IP rule set

1. Go to **Rules > IP Rules > Add > IP Rule**
2. Under **General** enter:
   - **Name:** Enter a suitable name such as *nat_pool_rule*
   - **Action:** NAT
3. Under **Address filter** enter:
   - **Source Interface:** int
   - **Source Network:** int-net
   - **Destination Interface:** wan
   - **Destination Network:** all-nets
   - **Service:** HTTP
4. Select the **NAT** tab and enter:
   - Check the **Use NAT Pool** option
   - Select **stateful_natpool** from the drop-down list
5. Click **OK**

8.3. Static Address Translation

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* (SAT).

Unlike NAT, SAT requires more than just 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 CorePlus execute the SAT rule.

8.3.1. Translation of a Single IP Address (1:1)

The simplest form of SAT usage is translation of a single IP address. A very common scenario for this is to enable external users to access a protected server having a private address. This scenario is also sometimes referred to as a *Virtual IP* or *Virtual Server* in some other manufacturer's products.

Example 8.3. 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.

CLI

First, change the current category to be the *main* IP rule set:

```
Device:/> cc IPRuleSet main
```

Next, create a SAT IP rule:

```
Device:/main> add IPRule Action=SAT Service=http SourceInterface=any
     SourceNetwork=all-nets DestinationInterface=core
     DestinationNetwork=ip_wan SATTranslate=DestinationIP
     SATTranslateToIP=10.10.10.5 Name=SAT_HTTP_To_DMZ
```

Then create a corresponding Allow rule:

```
Device:/main> add IPRule action=Allow Service=http SourceInterface=any
     SourceNetwork=all-nets DestinationInterface=core
     DestinationNetwork=ip_wan Name=Allow_HTTP_To_DMZ
```

Web Interface

First create a SAT rule:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Specify a suitable name for the rule, for example *SAT_HTTP_To_DMZ*
3. Now enter:
   - **Action:** SAT
   - **Service:** http
   - **Source Interface:** any
   - **Source Network:** all-nets
   - **Destination Interface:** core
   - **Destination Network:** ip_wan

4. Under the **SAT** tab, make sure that the **Destination IP Address** option is selected
5. In the **New IP Address** textbox, enter *10.10.10.5*
6. Click **OK**

Then create a corresponding Allow rule:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Specify a suitable name for the rule, for example *Allow_HTTP_To_DMZ*
3. Now enter:
   - **Action:** Allow
   - **Service:** http
   - **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** 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?

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. 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 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 8.4. 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

The order of events is as follows:

- 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 example, 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	lannet	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**

In 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* and this would completely avoid all the problems associated with address translation. However, this is not always practical.

8.3.2. 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 where the first original IP address will be translated to the first IP address in the translation list and so on.

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 8.5. 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.

CLI

Create an address object for the public IP addresses:

```
Device:/> add Address IP4Address wwwsrv_pub Address=195.55.66.77-195.55.66.81
```

Now, create another object for the base of the web server IP addresses:

```
Device:/> add Address IP4Address wwwsrv_priv_base Address=10.10.10.5
```

Publish the public IP addresses on the wan interface using ARP publish. One ARP item is needed for every IP address:

```
Device:/> add ARP Interface=wan IP=195.55.66.77 mode=Publish
```

Repeat this for all the five public IP addresses.

Next, change the current category to be the *main* IP rule set:

```
Device:/> cc IPRuleSet main
```

Next, create a SAT rule for the translation:

```
Device:/main> add IPRule Action=SAT Service=http SourceInterface=any
     SourceNetwork=all-nets DestinationInterface=core
     DestinationNetwork=wwwsrv_pub SATTranslateToIP=wwwsrv_priv_base
     SATTranslate=DestinationIP
```

Finally, create a corresponding Allow Rule:

```
Device:/main> add IPRule Action=Allow Service=http SourceInterface=any
     SourceNetwork=all-nets DestinationInterface=core
     DestinationNetwork=wwwsrv_pub
```

Web Interface

Create an address object for the public IP address:

1. Go to **Objects > Address Book > Add > IP address**
2. Specify a suitable name for the object, for example *wwwsrv_pub*

3. Enter *195.55.66.77 - 195.55.66.77.81* as the **IP Address**
4. Click **OK**

Now, create another address object for the base of the web server IP addresses:

1. Go to **Objects > Address Book > Add > IP address**
2. Specify a suitable name for the object, for example *wwwsrv_priv_base*
3. Enter *10.10.10.5* as the **IP Address**
4. Click **OK**

Publish the public addresses in the wan interface using ARP publish. One ARP item is needed for every IP address:

1. Go to **Interfaces > ARP > Add > ARP**
2. Now enter:
   - **Mode:** Publish
   - **Interface:** wan
   - **IP Address:** 195.55.66.77
3. Click **OK** and repeat for all 5 public IP addresses

Create a SAT rule for the translation:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Specify a suitable name for the rule, for example *SAT_HTTP_To_DMZ*
3. Now enter:
   - **Action:** SAT
   - **Servce:** http
   - **Source Interface:**any
   - **Source Network:** all-nets
   - **Destination Interface:** core
   - **Destination Network:** wwwsrv_pub
4. Switch to the **SAT** tab
5. Make sure that the **Destination IP Address** option is selected
6. In the **New IP Address** dropdown list, select *wwwsrv_priv*
7. Click **OK**

Finally, create a corresponding Allow Rule:

1. Go to **Rules > IP Rules > Add > IPRule**
2. Specify a suitable name for the rule, for example *Allow_HTTP_To_DMZ*
3. Now enter:
   - **Action:** Allow
   - **Service:** http
   - **Source Interface:**any
   - **Source Network:** all-nets
   - **Destination Interface:** core

- **Destination Network:** wwwsrv_pub

4. Click **OK**

8.3.3. 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 all-nets is the destination, All-to-One mapping is always done.

8.3.4. Port Translation

Port Translation, also known as *Port Address Translation* (PAT), 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.

8.3.5. Protocols handled by SAT

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 security 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 7.2, “ALGs”.

8.3.6. Multiple SAT rule matches

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 CorePlus 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.

8.3.7. 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 now 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 is as follows:

- 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.
- Return traffic from wwwsrv:80 will match rules 2 and 3.
- 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 Clavister Security Gateway.
- Return traffic will automatically be handled by the Clavister Security Gateway's stateful

inspection mechanism.

Chapter 9. User Authentication

This chapter describes how CorePlus implements user authentication.

9.1. Overview

In situations where individual users connect to protected resources through the Clavister Security Gateway, the administrator will often require that each user goes through a process of *authentication* before access is allowed. This chapter deals with setting up authentication for CorePlus but first the general issues involved in authentication are examined.

Proving Identity

The aim of authentication is to have the user prove their identity so that the network administrator can allow or deny access to resources based on that identity. Possible types of proof could be:

A. Something the user is. Unique attributes that are different for every person, such as a fingerprint.

B. Something the user has, such a passcard, a X.507 Digital Certificate or Public and Private Keys.

C. Something the user knows such as a password.

Method **A** may require a special biometric reader. Another problem is that the feature often cannot be replaced if it is lost. Methods **B** and **C** are therefore the most common in network security. However, these have drawbacks: keys might be intercepted, passcards might be stolen, passwords might be guessable, or people may simply be bad at keeping a secret. Methods **B** and **C** are sometimes combined, for example in a passcard that requires a password or pincode for use.

Using Username/Passwords

This chapter deals specifically with user authentication through validation of username/password combinations manually entered by a user attempting to gain access to resources. Access to the Internet using the HTTP protocol through the Clavister Security Gateway is an example of this, where a username/password combination is the primary authentication method.

In using this approach, passwords are often subject to attacks by guesswork or systematic searches. To counter this, a password should be carefully chosen. Ideally it should:

- Be more than 8 characters with no repeats.
- Use random character sequences not commonly found in phrases.
- Contain both lower and upper case alphabetic characters.
- Contain both digits and special characters.

To remain secure, passwords should also:

- Not be recorded anywhere in written form.
- Never be revealed to anyone else.

- Changed on a regular basis such as every three months.

9.2. Authentication Setup

9.2.1. Setup Summary

The following list summarizes the steps for User Authentication setup with CorePlus:

- Set up a database of users, each with a username/password combination. This can exist locally in a CorePlus *User DB* object, or remotely on a RADIUS server and will be designated as the *Authentication Source*. Membership of an *Authentication Group* can optionally be specified for each user.
- Define a *User Authentication Rule* which describes which traffic is to be authenticated and which *Authentication Source* will be used.
- Define an IP object for the IP addresses of the clients that will be authenticated. Associate this with an Authentication Group if required.
- Set up IP rules to allow the authentication to take place and also to allow access to resources by the clients belonging to the IP object set up in the previous step.

The following sections describe the components of these steps in detail.

Authentication Sources

The database that an Authentication Rule uses to check a user's username/password combination can be one of the following types:

- A *Local Database* internal to CorePlus that is set up by the administrator.
- A RADIUS server which is external to the Clavister Security Gateway.

9.2.2. Local User Databases

Local User Databases are administrator defined registries internal to CorePlus which contain the profiles of authorized users and user groups. Combinations of usernames/password can be entered into these with passwords stored using reversible cryptography for security.

Group Membership

Each user entered into the Local Database can optionally be specified to be a member of one or more *Groups*. Authentication Groups are not pre-defined but rather entered as text strings. These text strings are case sensitive and must always be entered in exactly the same way. Authentication Groups have no relationship with Authentication Rules but are instead associated with IP objects which are then used in the IP rule set.

When specifying the Source Network for an IP rule, a user defined IP object can be used and a group can be associated with that IP object. This will mean that the IP rule will then only apply to logged-in clients who also belong to the Source Network's group.

The purpose of this is to restrict access to certain networks to a particular group by having IP rules which will only apply to members of that group. To gain access to a resource there must be an IP rule that allows it and the client must belong to the same group as the rule's Source Network group.

9.2.3. External RADIUS Servers

Reasons for External Servers

In a larger network topology with a larger administration workload, it is often preferable to have a central authentication database on a dedicated server. When there is more than one Clavister Security Gateway in the network and thousands of users, maintaining separate authentication databases on each device becomes problematic. Instead, an external authentication server can validate username/password combinations by responding to requests from CorePlus. To provide this, CorePlus supports the *Remote Authentication Dial-in User Service* (RADIUS) protocol.

RADIUS with CorePlus

CorePlus acts as a RADIUS client, sending user credentials and connection parameter information as a RADIUS message to a nominated RADIUS server. The server processes the requests and sends back a RADIUS message to accept or deny them. One or more external servers can be defined in CorePlus.

RADIUS Security

To provide security, a common *shared secret* is configured on both the RADIUS client and the server. This secret enables encryption of the messages sent from the RADIUS client to the server and is commonly configured as a relatively long text string. The string can contain up to 100 characters and is case sensitive.

RADIUS uses PPP to transfer username/password requests between client and RADIUS server, as well as using PPP authentication schemes such as PAP and CHAP. RADIUS messages are sent as UDP messages via UDP port 1812.

9.2.4. Authentication Rules

Authentication Rules are set up in a way that is similar to other CorePlus security policies, by specifying which traffic is to be subject to the rule. They differ from other policies in that the destination network/interface is not of interest but only the source network/interface. An Authentication Rule has the following parameters:

- **Interface** - The source interface on which the connections to be authenticated will arrive.
- **Source IP** - The source network from which these connections will arrive.
- **Authentication Source** - This specifies that authentication is to be done against a **Local** database defined within CorePlus or by using a **RADIUS** server (discussed in detail below).

  A further option, **Disallow**, can be used so that a negative rule can be created which says "never authenticate given these conditions". This option might be used, for instance, to never authenticate connections comming in on a particular interface. Typically, *Disallow* rules are located at the end of the authentication rule set.
- **Agent** - The type of traffic being authenticated. This can one of:
  - **HTTP** or **HTTPS** - Web connections to be authenticated via a pre-defined or custom web page (see the detailed HTTP explanation below).
  - **PPP** - L2TP or PPP tunnel authentication.
  - **XAUTH** - IKE authentication which is part of IPsec tunnel establishment.

Connection Timeouts

An Authentication Rule can specify the following timeouts related to a user session:

- **Idle Timeout** - How long a connection is idle before being automatically terminated (1800 seconds by default).

- **Session Timeout** - The maximum time that a connection can exist (no value is specified by default).

If an authentication server is being used then the option to **Use timeouts received from the authentication server** can be enabled to have these values set from the server.

Multiple Logins

An Authentication Rule can specify how *multiple logins* are handled where more than one user from different source IP addresses try to login with the same username. The possible options are:

- Allow multiple logins so that more than one client can use the same username/password combination.
- Allow only one login per username.
- Allow one login per username and logout an existing user with the same name if they have been idle for a specific length of time when the new login occurs.

9.2.5. Authentication Processing

The list below describes the processing flow through CorePlus for username/password authentication:

1. A user creates a new connection to the Clavister Security Gateway.
2. CorePlus sees the new user connection on an interface and checks the *Authentication rule set* to see if there is a matching rule for traffic on this interface, coming from this network and data which is one of the following types:
   - HTTP traffic
   - HTTPS traffic
   - IPsec tunnel traffic
   - L2TP tunnel traffic
   - PPTP tunnel traffic
3. If no Authentication Rule matches, the connection is allowed if the IP rule set permits it and nothing further happens in the authentication process.
4. Based on the settings of the matching authentication rule, CorePlus prompts the user with an authentication request.
5. The user replies by entering their identification information which is usually a username/password pair.
6. CorePlus validates the information against the *Authentication Source* specified in the authentication rule. This will be either a local CorePlus database or an external RADIUS database server.
7. CorePlus then allows further traffic through this connection as long as authentication was successful and the service requested is allowed by a rule in the IP rule set. That rule's Source Network object has either the **No Defined Credentials** option enabled or alternatively it is associated with a group and the user is also a member of that group.
8. If a timeout restriction is specified in the authentication rule then the authenticated user will be automatically logged out after that length of time without activity.

Any packets from an IP address that fails authentication are discarded (unless they are caught be another rule).

9.2.6. A Group Usage Example

To illustrate Authentication Group usage, lets suppose that there are a set of users which will login from a network 192.168.1.0/24 connected to the *lan* interface. The requirement is to restrict access to a network called *important_net* on the *int* interface to one group of trusted users, while the other less-trusted users can only access another network called *regular_net* on the *dmz* interface.

Assuming we have a CorePlus internal database already defined, we add the users to this database with appropriate username/password pairs and a specific **Group** string. One set of users would be assigned to the group *trusted* and the other to the group *untrusted*.

We now define two IP objects for the same network 192.168.1.0/24. One IP object is called *untrusted_net* and has its **Group** parameter set to the string *untrusted*. The other IP object is called *trusted_net* and its **Group** parameter is set to the string *untrusted*.

The final step is to set up the rules in the IP rule set as shown below:

	Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
1	Allow	lan	trusted_net	int	important_net	All
2	Allow	lan	untrusted_net	dmz	regular_net	All

If we wanted to allow the *trusted* group users to also be able to access the regular network we could add a third rule to permit this:

	Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
1	Allow	lan	trusted_net	int	important_net	All
2	Allow	lan	trusted_net	dmz	regular_net	All
3	Allow	int	untrusted_net	dmz	regular_net	All

9.2.7. HTTP Authentication

Where users are communicating through a web browser using the HTTP protocol then authentication can be done by presenting the user with HTML pages to retrieve required user information. This is sometimes referred to as *WebAuth* and the setup requires further considerations.

Changing the Management WebUI Port

HTTP authentication will collide with the WebUI's remote management service which also uses TCP port 80. To avoid this, the WebUI port number should be changed before configuring authentication. Do this by going to **Remote Management > advanced settings** in the WebUI and changing the setting **WebUI HTTP Port**. Port number 81 could instead, be used for this setting.

Agent Options

For HTTP and HTTPS authentication there is a set of options in Authentication Rules called **Agent Options**. These are:

- **Login Type** - This can be one of:
  - **FORM** - The user is presented with an HTML page for authentication which is filled in and the data sent back to CorePlus with a POST. An HTML pre-defined in CorePlus will be used but this can be customized as described below.
  - **BASICAUTH** - This sends a **401 - Authentication Required** message back to the browser which will cause it to use its own inbuilt dialog to ask the user for a username/password

combination. A **Realm String** can optionally be specified which will appear in the browser's dialog.

FORM is recommended over **BASICAUTH** because in some cases the browser might hold the login data in its cache.

- If the **Agent** is set to *HTTPS* then the **Host Certificate** and **Root Certificate** have to be chosen from a list of certificates already loaded into CorePlus.

Setting Up IP Rules

HTTP authentication cannot operate unless a rule is added to the IP rule set to explicitly allow authentication to take place. If we consider the example of a number of clients on the local network *lannet* who would like access to the public Internet on the *wan* interface then the IP rule set would contain the following rules.

	Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
1	Allow	lan	lannet	core	ip_lan	http-all
2	NAT	lan	trusted_users	wan	all-nets	http-all
3	NAT	lan	lannet	wan	all-nets	dns-all

The first rule allows the authentication process to take place and assumes the client is trying to access the *ip_lan* IP address, which is the IP address of the interface on the Clavister Security Gateway where the local network connects.

The second rule allows normal surfing activity but we cannot just use *lannet* as the source network since the rule would trigger for any unauthenticated client from that network. Instead, the source network is an administrator defined IP object called *trusted_users* which is the same network as *lannet* but has additionally either the Authentication option **No Defined Credentials** enabled *or* has an Authentication Group assigned to it (which is the same group as that assigned to the users).

The third rule allows DNS lookup of URLs.

Forcing Users to a Login Page

With this setup, when users that are not authenticated try to surf to any IP except *ip_lan* they will fall through the rules and their packets will be dropped. To always have these users come to the authentication page we must add a **SAT** rule and its associated **Allow** rule. The rule set will now look like this:

	Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
1	Allow	lan	lannet	core	ip_lan	http-all
2	NAT	lan	trusted_users	wan	all-nets	http-all
3	NAT	lan	lannet	wan	all-nets	dns-all
4	SAT	lan	lannet	wan	all-nets all-to-one 127.0.0.1	http-all
5	Allow	lan	lannet	wan	all-nets	http-all

The **SAT** rule catches all unauthenticated requests and must be set up with an all-to-one address mapping that directs them to the address *127.0.0.1* which corresponds to **core** (CorePlus itself).

Example 9.1. 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.

Web Interface

Step A

1. Go to **User Authentication > Local User Databases > Add > LocalUserDatabase**
2. Now enter:
   - **Name:** lannet_auth_users
   - **Comments:** folder for "lannet" authentication user group - "users"
3. Click **OK**

Step B

1. Go to **lannet_auth_users > Add > User**
2. Now enter:
   - **Username:** Enter the user's account name, for example *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 a comma - *users* for this example
3. Click **OK**
4. Repeat **Step B** to add all the *lannet* users having the membership of *users* group into the *lannet_auth_users* folder

Example 9.2. Configuring a RADIUS Server

The following steps illustrate how a RADIUS server is typically configured.

Web Interface

1. **User Authentication > External User Databases> Add > External User Database**
2. Now enter:
   a. **Name:** Enter a name for the server, for example *ex-users*
   b. **Type:** Select RADIUS
   c. **IP Address:** Enter the IP address of the server, or enter the symbolic name if the server has been defined in the **Address Book**
   d. **Port:** 1812 (RADIUS service uses UDP port 1812 by default)
   e. **Retry Timeout:** 2 (CorePlus will resend the authentication request to the sever if there is no response after the timeout, for example every 2 seconds. This will be retried a maximum of 3 times)
   f. **Shared Secret:** Enter a text string here for basic encryption of the RADIUS messages
   g. **Confirm Secret:** Retype the string to confirm the one typed above
3. Click **OK**

9.3. Customizing HTML Pages

User Authentication makes use of a set of HTML files to present information to the user during the authentication process such as when a login fails. These web pages, sometimes referred to as *HTTP banner files*, are stored within CorePlus but can be customized to suit a particular installation's needs. The available files are:

FormLogin
LoginSuccess
LoginFailure
LoginAlreadyDone
LoginChallenge
LoginChallengeTimeout
LoginSuccess
LoginSuccessBasicAuth
LoginFailure
FileNotFound

The WebUI provides a simple way to download, edit and upload the edited HTML to CorePlus. The description of doing this given below mirrors the procedure described in Section 7.3.4.4, “Customizing HTML Pages”.

To perform customization it is necessary to first create a new, named **Auth Banner Files** object. This new object automatically contains a copy of all the files in the *Default* Auth Banner Files object. These new files can then be edited and uploaded back to CorePlus. The original *Default* object cannot be edited. The following example goes through the steps for customization.

Example 9.3. Editing Content Filtering HTTP Banner Files

This example shows how to modify the contents of the *URL forbidden* HTML page.

Web Interface

1. Go to **Objects > HTTP Banner files > Add > Auth Banner Files**
2. Enter a name such as *new_forbidden* and press **OK**
3. The dialog for the new set of ALG banner files will appear
4. Click the **Edit & Preview tab**
5. Select *FormLogin* from the **Page** list
6. Now edit the HTML source that appears in the text box for the Forbidden URL page
7. Use **Preview** to check the layout if required
8. Press **Save** to save the changes
9. Click **OK** to exit editing
10. Go to **Objects > ALG** and select the relevant HTML ALG
11. Select *new_forbidden* as the **HTML Banner**
12. Click **OK**
13. Go to **Configuration > Save & Activate** to activate the new file

Note
*In the above example, more than one HTML file can be edited in a session but the **Save** button should be pressed to save any edits before beginning editing on another file.*

Uploading with SCP

It is possible to upload new HTTP Banner files using SCP. The steps to do this are:

1. Since SCP cannot be used to download the original default HTML, the source code must be first copied from the WebUI and pasted into a local text file which is then edited using an appropriate editor.

2. A new **Auth Banner Files** object must exist which the edited file(s) is uploaded to. If the object is called *ua_html*, the CLI command to create this object is:

   ```
   Device:/> add HTTPAuthBanners ua_html
   ```

   This creates an object which contains a copy of all the *Default* user auth banner files.

3. The modified file is then uploaded using SCP. It is uploaded to the object type *HTTPAuthBanner* and the object *ua_html* with property name *FormLogin*. If the edited *Formlogon* local file is called *my.html* then using the Open SSH SCP client, the upload command would be:

   ```
   pscp my.html admin@10.5.62.11:HTTPAuthBanners/ua_html/FormLogin
   ```

   The usage of SCP clients is explained further in Section 3.1.6, “Secure Copy”.

4. Using the CLI, the relevant user authentication rule should now be set to use the *ua_html*. If the rule us called *my_auth_rule*, the command would be:

   ```
   set UserAuthRule my_auth_rule HTTPBanners=ua_html
   ```

5. As usual, use the *activate* followed by the *commit* CLI commands to activate the changes on the Clavister Security Gateway.

HTML Page Parameters

The HTML pages contain a number of parameters that can be used where it is appropriate. The parameters available are:

- **%URL%** - The URL which was requested
- **%IPADDR%** - The IP address which is being browsed from
- **%REASON%** - The reason that access was denied

Chapter 10. VPN

This chapter describes the *Virtual Private Network* (VPN) functionality in CorePlus.

10.1. Overview

10.1.1. VPN Usage

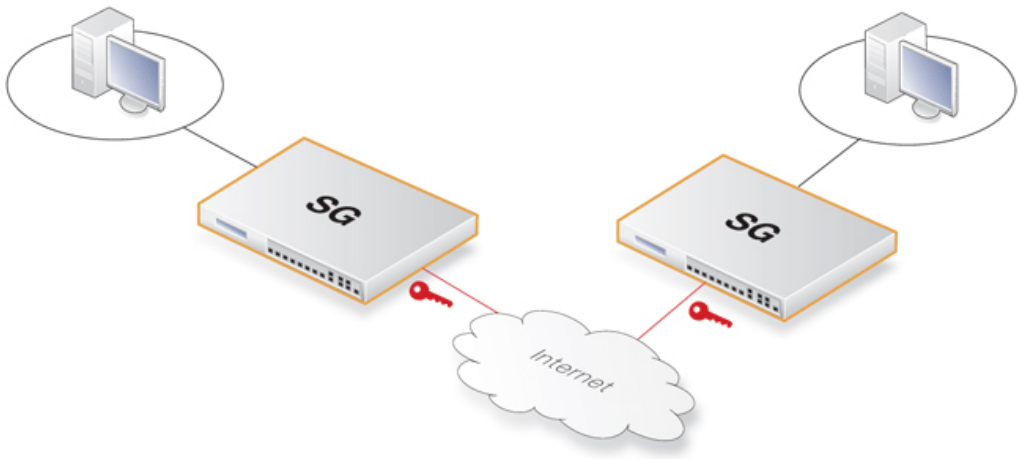
The Internet is increasingly used as a means to connect together computers since it offers efficient and inexpensive communication. The requirement therefore exists for data to traverse the Internet to its intended recipient without another party being able to read or alter it.

It is equally important that the recipient can verify that no one is falsifying data, in other words, pretending to be someone else. *Virtual Private Networks* (VPNs) meet this need, providing a highly cost effective means of establishing secure links between two co-operating computers so that data can be exchanged in a secure manner.

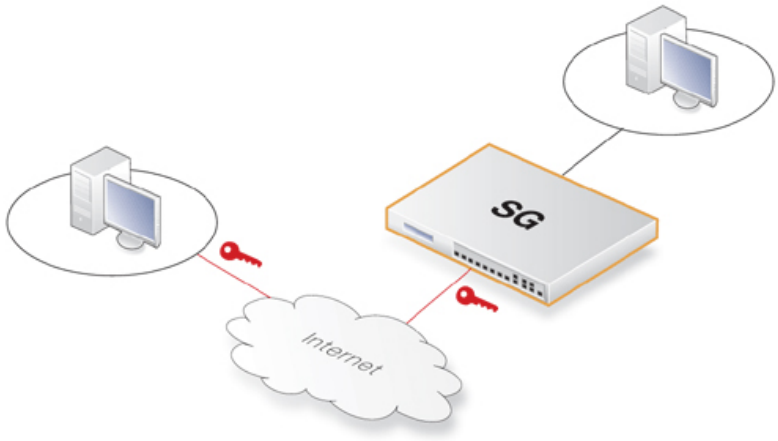
VPN allows the setting up of a *tunnel* between two devices known as *tunnel endpoints*. All data flowing through the tunnel is then secure. The mechanism that provides tunnel security is *encryption*.

There are two common scenarios where VPN is used:

1. **LAN to LAN connection** - Where two internal networks need to be connected together over the internet. In this case, each network is protected by an individual Clavister Security Gateway and the VPN tunnel is set up between them.



2. **Client to LAN connection** - Where many remote clients need to connect to an internal network over the internet. In this case, the internal network is protected by the Clavister Security Gateway to which the client connects and the VPN tunnel is set up between them.



10.1.2. VPN Encryption

Encryption of VPN traffic is done using the science of *cryptography*. Cryptography is an umbrella expression covering 3 techniques and benefits:

Confidentiality	No one but the intended recipients is able to receive and understand the communication. Confidentiality is accomplished by encryption.
Authentication and Integrity	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.
Non-repudiation	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.

10.1.3. VPN Planning

An attacker targeting 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 at 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 easier.

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

Endpoint Security

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.

Placement in a DMZ

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 and CorePlus VPN has this feature.

10.1.4. Key Distribution

Key distribution schemes are best planned in advance. Issues that need to be addressed include:

- How will keys be distributed? Email is not good. Phone conversations might be secure enough.
- How many different keys should be used? One key per user? One per group of users? One per LAN-to-LAN connection? One key for all users and one key for all LAN-to-LAN connections? It is probably better using more keys than is necessary today since it will be 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?

10.2. VPN Quick Start

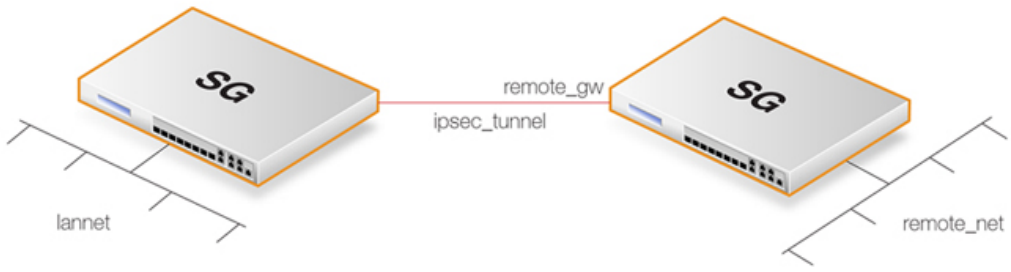
Later sections in this chapter will explore VPN components in detail. To help put those later sections in context, this section is a quick start summary of the key steps in VPN setup.

It outlines the individual steps in setting up VPNs for the most common VPN scenarios. These are:

- IPsec LAN to LAN with Pre-shared Keys
- IPsec LAN to LAN with Certificates
- IPsec Roaming Clients with Pre-shared Keys
- IPsec Roaming Clients with Certificates
- L2TP Roaming Clients with Pre-Shared Keys
- L2TP Roaming Clients with Certificates
- PPTP Roaming Clients

10.2.1. IPsec LAN to LAN with Pre-shared Keys

1. Create a **Pre-shared Key** object.
2. Optionally create a new **IKE Algorithms** object and/or an **IPsec Algorithms** object if the default algorithm proposal lists do not provide a set of algorithms that are acceptable to the tunnel remote end point. This will depend on the capabilities of the device at the other end of the VPN tunnel.
3. In the **Address Book** create IP objects for:
   - The remote VPN gateway which is the IP address of the network device at the other end of the tunnel (let's call this object *remote_gw*).
   - The remote network which lies behind the remote VPN gateway (let's call this object *remote_net*).
   - The local network behind the Clavister Security Gateway which will communicate across the tunnel. Here we will assume that this is the pre-defined address *lannet* and this network is attached to the CorePlus *lan* interface.



4. Create an **IPsec Tunnel** object (let's call this object *ipsec_tunnel*). Specify the following tunnel parameters:
   - Set **Local Network** to *lannet*.
   - Set **Remote Network** to *remote_net*.

- Set **Remote Endpoint** to *remote_gw*.
- Set **Encapsulation mode** to *Tunnel*.
- Choose the IKE and IPsec algorithm proposal lists to be used.
- For **Authentication** select the **Pre-shared Key** object defined in step **(1)** above.

The **IPsec Tunnel** object can be treated exactly like any CorePlus *Interface* object in later steps.

5. Set up two IP rules in the IP rule set for the tunnel:
   - An **Allow** rule for outbound traffic that has the previously defined *ipsec_tunnel* object as the **Destination Interface**. The rule's **Destination Network** is the remote network *remote_net*.
   - An **Allow** rule for inbound traffic that has the previously defined *ipsec_tunnel* object as the *Source Interface*. The **Source Network** is *remote_net*.

Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
Allow	lan	lannet	ipsec_tunnel	remote_net	All
Allow	ipsec_tunnel	remote_net	lan	lannet	All

The Service used in these rules is *All* but it could be a predefined service.

6. Define a new CorePlus **Route** which specifies that the VPN Tunnel *ipsec_tunnel* is the Interface to use for routing packets bound for the remote network at the other end of the tunnel.

Interface	Network	Gateway
ipsec_tunnel	remote_net	

10.2.2. IPsec LAN to LAN with Certificates

LAN to LAN security is usually provided with pre-shared keys but sometimes it may be desirable to use X.509 certificates instead. If this is the case, Certificate Authority (CA) signed certificates may be used and these come from an internal CA server or from a commercial supplier of certificates. Alternatively, self-signed certificates can be used and these can be generated from a number of utilities downloadable from the internet.

Creating a LAN to LAN tunnel with certificates follows exactly the same procedures as the previous section except that certificates replace pre-shared keys for authentication.

Two certificates are required for a LAN to LAN tunnel and both should be signed by the same CA. The same two are used at either end of an IPsec tunnel but their use is reversed at either end. In other words: one certificate is used as the *root certificate* at one tunnel end, call it **Side A**, and as the *host certificate* at the other end, call it **Side B**. The second certificate is used in the opposite way: it is the *host certificate* at **Side A** and the *root certificate* at **Side B**.

The certificate setup steps are:

1. The CorePlus date and time must be set correctly since certificates can expire.
2. Open the WebUI management interface for the Clavister Security Gateway at one end of the tunnel.
3. Under **Authentication Objects**, add the *Root Certificate* and *Host Certificate* into CorePlus.

The root certificate needs to have 2 parts added: a certificate file and a private key file. The gateway certificate needs just the certificate file added.

4. Set up the **IPsec Tunnel** object as for pre-shared keys, but specify the certificates to use under **Authentication**. Do this with the following steps:

   a. Enable the **X.509 Certificate** option.

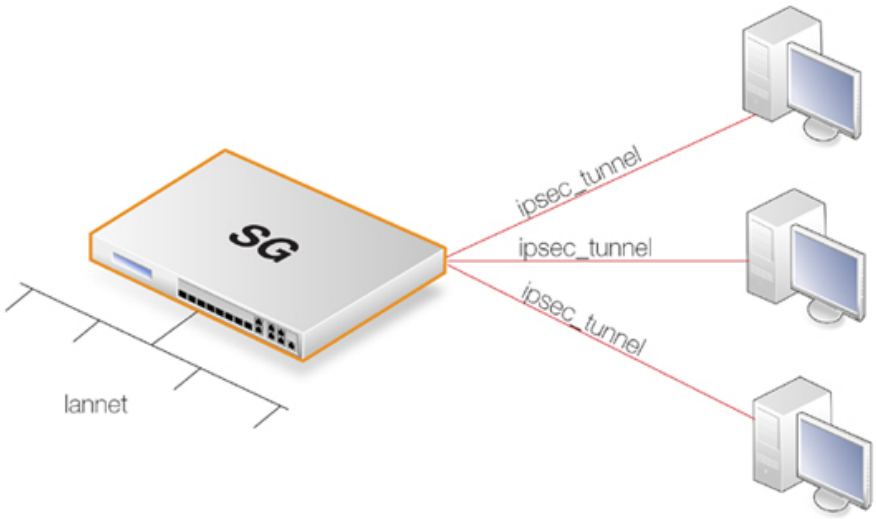
   b. Add the **Root Certificate** to use.

   c. Select the **Gateway Certificate**.

5. Open the WebUI management interface for the Clavister Security Gateway at the other side of the tunnel and repeat the above steps but reversing the certificate usage. What was the root certificate is now added as the gateway certificate, and its private key file is not used. What was the gateway certificate is now added as the root certificate and its private key file will be used.

Also review Section 10.6, “CA Server Access” below, which describes important considerations for certificate validation.

10.2.3. IPsec Roaming Clients with Pre-shared Keys



This section details the setup with roaming clients connecting through an IPsec tunnel with pre-shared keys. There are two types of roaming clients:

A. The IP addresses of the clients are already allocated.

B. The IP addresses of clients are not known beforehand and must be handed out by CorePlus as the clients connect.

A. IP addresses already allocated

The IP addresses may be known beforehand and have been pre-allocated to the roaming clients before they connect. The client's IP address will be manually input into the VPN client software.

1. Set up user authentication. *XAuth* user authentication is not required with IPsec roaming clients but is recommended (this step could initially be left out to simplify setup). The authentication source can be one of the following:

- A **Local User DB** object which is internal to CorePlus.
- An external authentication server.

An internal user database is easier to set up and is assumed here. Changing this to an external server is simple to do later.

To implement user authentication with an internal database:

- Define a **Local User DB** object (let's call this object *TrustedUsers*).
- Add individual users to *TrustedUsers*. This should consist of at least a username and password combination.

  The **Group** string for a user can be specified if its group's access is to be restricted to certain source networks. **Group** can be specified (with the same text string) in the **Authentication** section of an IP object. If that IP object is then used as the **Source Network** of a rule in the IP rule set, that rule will only apply to a user if their **Group** string matches the **Group** string of the IP object. (note: **Group** has no meaning in **Authentication Rule**s).
- Create a new **User Authentication Rule** with the **Authentication Source** set to *TrustedUsers*. The other parameters for the rule are:

Agent	Auth Source	Src Network	Interface	Client Source IP
XAUTH	Local	all-nets	any	all-nets (0.0.0.0/0)

2. The **IPsec Tunnel** object *ipsec_tunnel* should have the following parameters:
   - Set **Local Network** to *lannet*.
   - Set **Remote Network** to *all-nets*
   - Set **Remote Endpoint** to *all-nets*.
   - Set **Encapsulation mode** to *Tunnel*.
   - Set the IKE and IPsec algorithm proposal lists to match the capabilities of the clients.
   - No routes can be predefined so the option **Dynamically add route to the remote network when tunnel established** should be enabled for the tunnel object. If *all-nets* is the destination network, the option *Add route for remote network* should be disabled.
   - Enable the option **Require IKE XAuth user authentication for inbound IPsec tunnels**. This will enable a search for the first matching XAUTH rule in the authentication rules.
3. The IP rule set should contain the single rule:

Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
Allow	ipsec_tunnel	all-nets	lan	lannet	All

Once an **Allow** rule permits the connection to be set up, bidirectional traffic flow is allowed which is why only one rule is used here. Instead of *all-nets* being used in the above, a more secure defined IP object could be used which specifies the exact range of the pre-allocated IP addresses.

B. IP addresses handed out by CorePlus

If the client IP addresses are not known then they must be handed out by CorePlus. To do this the

above must be modified with the following:

1. If a specific IP address range is to be used as a pool of available addresses then:
   - Create a **Config Mode Pool** object (there can only be one associated with a CorePlus installation) and in it specify the address range.
   - Enable the **IKE Config Mode** option in the **IPsec Tunnel** object *ipsec_tunnel*.
2. If client IP addresses are to be retrieved through DHCP:
   - Create an **IP Pool** object and in it specify the DHCP server to use. The DHCP server can be specified as a simple IP address or alternatively as being accessible on a specific interface. If an internal DHCP server is to be used then specify the loopback address *127.0.0.1* as the DHCP server IP address.
   - Create a **Config Mode Pool** object (there can only be one associated with a CorePlus installation) and associate with it the IP Pool object defined in the previous step.
   - Enable the **IKE Config Mode** option in the **IPsec Tunnel** object *ipsec_tunnel*.

Configuring the IPsec Client

In both cases **(A)** and **(B)** above the IPsec client will need to configured with the URL of the Clavister Security Gateway as well as the pre-shared key.

10.2.4. IPsec Roaming Clients with Certificates

If certificates are used with IPsec roaming clients instead of pre-shared keys then no **Pre-shared Key** object is needed and the other differences in the setup described above are:

1. The CorePlus date and time must be set correctly since certificates can expire.
2. Load a *Gateway Certificate* and *Root Certificate* into CorePlus.
3. When setting up the **IPsec Tunnel** object, specify the certificates to use under **Authentication**. This is done by:
   a. Enable the **X.509 Certificate** option.
   b. Select the **Gateway Certificate**.
   c. Add the **Root Certificate** to use.
4. The IPsec client software will need to appropriately configured with the certificates and remote IP addresses.

The step to set up user authentication is optional since this is additional security to certificates.

Also review Section 10.6, “CA Server Access” below, which describes important considerations for certificate validation.

10.2.5. L2TP Roaming Clients with Pre-Shared Keys

Due to the inbuilt L2TP client in Microsoft Windows, L2TP is a popular choice for roaming client VPN scenarios. L2TP is usually encapsulated in IPsec to provide encryption with IPsec running in *transport mode* instead of *tunnel mode*. The steps for L2TP over IPsec setup are:

1. Create an IP object (let's call it *l2tp_pool*) which defines the range of IP addresses which can be

handed out to clients. The range chosen could be of two types:

- A range taken from the internal network to which clients will connect. If the internal network is 192.168.0.0/24 then we might use the address range 192.168.0.10 to 192.168.0.20. The danger here is that an IP address might be accidentally used on the internal network and handed out to a client.
- Use a new address range that is totally different to any internal network. This prevents any chance of an address in the range also being used on the internal network.

2. Define two other IP objects:
   - *ip_ext* which is the external public IP address through which clients connect (let's assume this is on the *ext* interface).
   - *ip_int* which is the internal IP address of the interface to which the internal network is connected (let's call this interface *int*).
3. Define a **Pre-shared Key** for the IPsec tunnel.
4. Define an *IPsec Tunnel* object (let's call this object *ipsec_tunnel*) with the following parameters:
   - Set **Local Network** to *ip_ext* (specify *all-nets* instead if CorePlus is behind a NATing device).
   - Set **Remote Network** to *all-nets*
   - Set **Remote Endpoint** to *none*
   - For **Authentication** select the **Pre-shared Key** object defined in the first step.
   - Set **Encapsulation Mode** to *Transport*.
   - Select the IKE and IPsec algorithm proposal lists to be used.
   - Enable the routing option **Dynamically add route to the remote network when tunnel established**. If *all-nets* is the destination network, the option *Add route for remote network* should be disabled.
5. Define an PPTP/L2TP Server object (let's call this object *l2tp_tunnel*) with the following parameters:
   - Set **Inner IP Address** to *ip_int*
   - Set **Tunnel Protocol** to *L2TP*
   - Set **Outer Interface Filter** to *ipsec_tunnel*
   - Set **Outer Server IP** to *ip_ext*
   - Select the **Microsoft Point-to-Point Encryption** allowed. Since IPsec encryption is used this can be set to be *None* only, otherwise double encryption will degrade throughput.
   - Set **IP Pool** to *l2tp_pool*.
   - Enable Proxy ARP on the *int* interface to which the internal network is connected.
   - Make the interface a member of a specific routing table so that routes are automatically added to that table. Normally the *main* table is selected.
6. For user authentication:
   - Define a **Local User DB** object (let's call this object *TrustedUsers*).

- Add individual users to *TrustedUsers*. This should consist of at least a username and password combination.

  The **Group** string for a user can also be specified. This is explained in the same step in the *IPsec Roaming Clients* section above.

- Define a User Authentication Rule:

Agent	Auth Source	Src Network	Interface	Client Source IP
PPP	Local	all-nets	l2tp_tunnel	all-nets (0.0.0.0/0)

7. To allow traffic through the L2TP tunnel the following rules should be defined in the IP rule set:

Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
Allow	l2tp_tunnel	l2tp_pool	any	int_net	All
NAT	ipsec_tunnel	l2tp_pool	ext	all-nets	All

The second rule would be included to allow clients to surf the Internet via the *ext* interface on the Clavister Security Gateway. The client will be allocated a private internal IP address which must be NATed if connections are then made out to the public Internet via the Clavister Security Gateway.

8. Set up the client. Assuming Windows XP, the **Create new connection** option in **Network Connections** should be selected to start the **New Connection Wizard**. The key information to enter in this wizard is: the resolvable URL of the Clavister Security Gateway or alternatively its *ip_ext* IP address.

   Then choose **Network** > **Properties**. In the dialog that opens choose the L2TP Tunnel and select **Properties**. In the new dialog that opens select the **Networking** tab and choose **Force to L2TP**. Now go back to the L2TP Tunnel properties, select the **Security** tab and click on the **IPsec Settings** button. Now enter the pre-shared key.

10.2.6. L2TP Roaming Clients with Certificates

If certificates are used with L2TP roaming clients instead of pre-shared keys then the differences in the setup described above are:

1. The CorePlus date and time must be set correctly since certificates can expire.

2. Load a *Gateway Certificate* and *Root Certificate* into CorePlus.

3. When setting up the **IPsec Tunnel** object, specify the certificates to use under **Authentication**. This is done by:

   a. Enable the **X.509 Certificate** option.

   b. Select the **Gateway Certificate.**

   c. Add the **Root Certificate** to use.

4. If using the Windows XP L2TP client, the appropriate certificates need to be imported into Windows before setting up the connection with the **New Connection Wizard**.

The step to set up user authentication is optional since this is additional security to certificates.

Also review Section 10.6, “CA Server Access” below, which describes important considerations for

certificate validation.

10.2.7. PPTP Roaming Clients

PPTP is simpler to set up than L2TP since IPsec is not used and instead relies on its own, less strong, encryption.

A major secondary disadvantage is not being able to NAT PPTP connections through a tunnel so multiple clients can use a single connection to the Clavister Security Gateway. If NATing is tried then only the first client that tries to connect will succeed.

The steps for PPTP setup are as follows:

1. In the **Address Book** define the following IP objects:
   - A *pptp_pool* IP object which is the range of internal IP addresses that will be handed out from an internal network.
   - An *int_net* object which is the internal network from which the addresses come.
   - An *ip_int* object which is the internal IP address of the interface connected to the internal network. let's assume this interface is *int*.
   - An *ip_ext* object which is the external public address which clients will connect to (let's assume this is on the *ext* interface).
2. Define a **PPTP/L2TP** object (let's call it *pptp_tunnel*) with the following parameters:
   - Set **Inner IP Address** to *ip_net*.
   - Set **Tunnel Protocol** to *PPTP*.
   - Set **Outer Interface Filter** to *ext*.
   - Set **Outer server IP** to *ip_ext*.
   - For **Microsoft Point-to-Point Encryption** it is recommended to disable all options except *128 bit* encryption.
   - Set **IP Pool** to *pptp_pool*
   - Enable Proxy ARP on the *int* interface.
   - As in L2TP, enable the insertion of new routes automatically into the *main* routing table.
3. Define a User Authentication Rule, this is almost identical to L2TP:

Agent	Auth Source	Src Network	Interface	Client Source IP
PPP	Local	all-nets	pptp_tunnel	all-nets (0.0.0.0/0)

4. Now set up the IP rules in the IP rule set:

Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
Allow	pptp_tunnel	pptp_pool	any	int_net	All
NAT	pptp_tunnel	pptp_pool	ext	all-nets	All

As described for L2TP, the **NAT** rule lets the clients access the public Internet via the Clavister Security Gateway.

5. Set up the client. For Windows XP, the procedure is exactly as described for L2TP above but without entering the pre-shared key.

10.3. IPsec Components

10.3.1. Overview

Internet Protocol Security (IPsec) is a set of protocols defined by the Internet Engineering Task Force (IETF) to provide IP security at the network layer. An IPsec based VPN is made up of 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.

10.3.2. Internet Key Exchange (IKE)

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 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

IKE and IPsec Lifetimes

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 crypto-analysis perspective.

The IPsec lifetime must be shorter than the IKE lifetime. The difference between the two must be a minimum of 5 minutes. 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.

It is recommended that the lifetime values are equal to or greater than the following values:

- IKE lifetime - 600 seconds (10 minutes).
- IPsec lifetime - 300 seconds (5 minutes).

If the administrator uses values which are less, CorePlus will accept them but a warning message will be presented when during the reconfiguration.

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 proposal 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 to 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 strongly 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

The *Local ID* is a piece of data representing the identity of the VPN gateway. With Pre-Shared Keys this is a unique piece of data uniquely identifying the tunnel endpoint.

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 *all-nets*, 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, for example for IPsec protected remote configuration.

This setting will typically be set to "tunnel" in most configurations.

Remote Endpoint

The remote endpoint (sometimes also referred to as the *remote gateway*) is the device that does the VPN decryption/authentication and that passes the unencrypted data on to its final destination. This field can also be set to *None*, forcing the Clavister Security Gateway to treat the remote address as the remote endpoint. 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.

The remote endpoint can be specified as a URL string such as *vpn.company.com*. If this is done, the prefix *dns:* must be used. The string above should therefore be specified as *dns:vpn.company.com*.

The remote endpoint is not used in transport mode.

Main/Aggressive Mode

The IKE negotiation has two modes of operation, main mode and aggressive mode.

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.

When using aggressive mode, some configuration parameters, such as Diffie-Hellman groups, and PFS, cannot be negotiated, resulting in a greater importance of having "compatible" configurations on both ends.

IPsec Protocols

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.

ESP provides encryption, authentication, or both. However, we do not recommend using encryption only, since it will dramatically decrease security.

More on ESP in ESP (Encapsulating Security Payload).

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.

More on AH in AH (Authentication Header).

Note
Clavister Security Gateways do not support AH.

IKE Encryption

This specifies the encryption algorithm used in the IKE negotiation, and depending on the algorithm, the size of the encryption key used.

The algorithms supported by CorePlus IPsec are:

- AES
- Blowfish
- 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 Group

This specifies the Diffie-Hellman group to use for the IKE exchange. The available DH groups are discussed below.

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. This value must be set greater than the IPsec SA lifetime.

PFS

With *Perfect Forwarding Secrecy* (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, 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.

PFS DH Group This specifies the DH group to use with PFS. This specifies the Diffie-Hellman group to use with PFS. The available DH groups are discussed below.

IPsec DH Group This specifies the Diffie-Hellman group to use for IPsec communication. The available DH groups are discussed below.

IPsec Encryption The encryption algorithm to use on the protected traffic.

This is not needed when AH is used, or when ESP is used without encryption.

The algorithms supported by Clavister Security Gateway VPNs are:

- AES
- Blowfish
- Twofish
- Cast128
- 3DES
- DES

IPsec Authentication This specifies the authentication algorithm used on the protected traffic.

This is not used when ESP is used without authentication, although it is not recommended to use ESP without authentication.

The algorithms supported by Clavister Security Gateway VPNs are:

- SHA1
- MD5

IPsec Lifetime 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. This value must be set lower than the IKE lifetime.

Diffie-Hellman Groups

Diffie-Hellman (DH) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications

channel through a series of exchanges. The DH protocol is used to establish shared secret keys for IKE, IPsec and PFS which are listed above.

The Diffie-Hellman groups supported by CorePlus are as follows:

- DH group 1 (768-bit)
- DH group 2 (1024-bit)
- DH group 5 (1536-bit)

These are available for IKE, IPsec and PFS. The security of DH key exchanges increases as the number of DH group bits become larger, as does the processing overhead involved in the DH exchange.

10.3.3. IKE Authentication

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.

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.

Manual Keying 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.

PSK

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.

PSK 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.

PSK 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 endpoint trusts.

Certificate 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.

Certificate 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.

10.3.4. IPsec Protocols (ESP/AH)

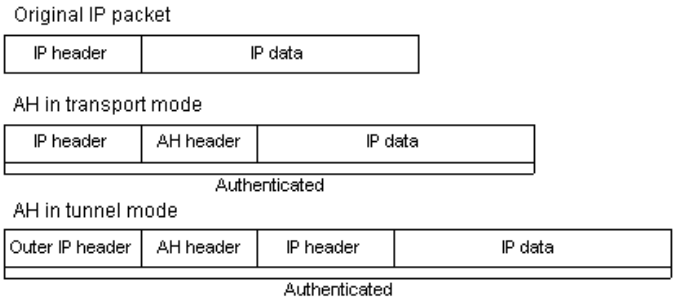
The IPsec protocols are the protocols used to protect the actual traffic being passed through the VPN. The actual protocols used and the keys used with those protocols are negotiated by IKE.

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.

Figure 10.1. The AH protocol



AH 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. 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.

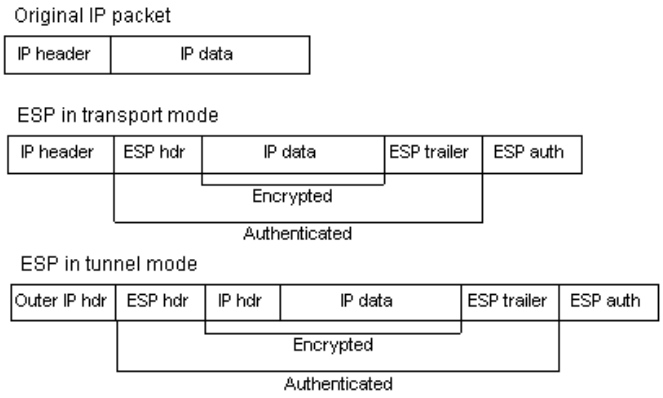
ESP (Encapsulating Security Payload)

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 10.2. The ESP protocol



10.3.5. Algorithm Proposal Lists

To agree on the VPN connection parameters, a negotiation process is performed. As a result of the negotiations, the IKE and IPsec *security associations* (SAs) are established. A *proposal list* of supported algorithms is the starting point for the negotiation. Each entry in the list defines parameters for a supported algorithm that the VPN tunnel end point device is capable of supporting (the shorter term *tunnel endpoint* will also be used in this manual). The initial negotiation attempts to agree on a set of algorithms that the devices at either end of the tunnel can support.

There are two types of proposal lists, IKE proposal lists and IPsec proposal lists. IKE lists are used during IKE Phase-1 (IKE Security Negotiation), while IPsec lists are using during IKE Phase-2 (IPsec Security Negotiation).

Several algorithm proposal lists are already defined by default in CorePlus for different VPN scenarios and user defined lists can be added.

Two IKE algorithm lists and two IPsec lists are already defined by default:

- **High**
  This consists of a more restricted set of algorithms to give higher security. The complete list is *3DES, AES, Blowfish, MD5, SHA1*.

- **Medium**
  This consists of a longer set of algorithms. The complete list is *3DES, AES, Blowfish, Twofish, CAST128, MD5, SHA1*.

Example 10.1. Using an Algorithm Proposal List

This example shows how to create and use an IPsec Algorithm 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.

CLI
First create a list of IPsec Algorithms:

```
Device:/> add IPsecAlgorithms esp-l2tptunnel DESEnabled=Yes DES3Enabled=Yes
     SHA1Enabled=Yes MD5Enabled=Yes
```

Then, apply the algorithm proposal list to the IPsec tunnel:

```
Device:/> set Interface IPsecTunnel MyIPsecTunnel IPsecAlgorithms=esp-l2tptunnel
```

Web Interface

First create a list of IPsec Algorithms:

1. Go to **Objects > VPN Objects > IPsec Algorithms > Add > IPsec Algorithms**
2. Enter a name for the list, for example *esp-l2tptunnel*
3. Now check the following:
   - **DES**
   - **3DES**
   - **SHA1**
   - **MD5**
4. Click **OK**

Then, apply the algorithm proposal list to the IPsec tunnel:

1. Go to **Interfaces > IPsec**
2. In the grid control, click the target IPsec tunnel
3. Select the recently created **esp-l2tptunnel** in the **IPsec Algorithms** control.
4. Click **OK**

10.3.6. 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 extremely vulnerable to dictionary attacks.

Pre-shared Keys can be generated automatically through the WebUI but they can also be generated through the CLI using the command *pskgen* (this command is fully documented in the *CLI Reference Guide*).

Beware of Non-ASCII Characters in a PSK on Different Platforms!

If a PSK is specified as a passphrase and not a hexadecimal value, the different encodings on different platforms can cause a problem with non-ASCII characters. Windows, for example, encodes pre-shared keys containing non ASCII characters in UTF-16 while CorePlus uses UTF-8. Even though they can seem the same at either end of the tunnel there will be a mismatch and this can sometimes cause problems when setting up a Windows L2TP client that connects to CorePlus.

Example 10.2. 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.

CLI
First create a Pre-shared Key. To generate the key automatically with a 64 bit (the default) key, use:

```
Device:/> pskgen MyPSK
```

To have a longer, more secure 512 bit key the command would be:

```
Device:/> pskgen MyPSK -size=512
```

Or alternatively, to add the Pre-shared Key manually, use:

```
Device:/> add PSK MyPSK Type=HEX PSKHex=<enter the key here>
```

Now apply the Pre-shared Key to the IPsec tunnel:

```
Device:/> set Interface IPsecTunnel MyIPsecTunnel PSK=MyPSK
```

Web Interface

First create a Pre-shared Key:

1. Go to **Objects > Authentication Objects > Add > Pre-shared key**
2. Enter a name for the pre-shared key, for example *MyPSK*
3. Choose **Hexadecimal Key** and click **Generate Random Key** to generate a key to the **Passphrase** textbox.
4. Click **OK**

Then, apply the pre-shared key to the IPsec tunnel:

1. Go to **Interfaces > IPsec**
2. In the grid control, click the target IPsec tunnel object
3. Under the **Authentication** tab, choose **Pre-shared Key** and select **MyPSK**
4. Click **OK**

10.3.7. Identification Lists

When certificates are used as authentication method for IPsec tunnels, the Clavister Security Gateway will accept all remote devices 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 example, 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 a certificate. Identification lists can thus be used to regulate what certificates that are given access to what IPsec tunnels.

Example 10.3. 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.

CLI

First create an Identification List:

```
Device:/> add IDList MyIDList
```

Then, create an ID:

```
Device:/> cc IDList MyIDList


Device:/MyIDList> add ID JohnDoe Type=DistinguishedName
     CommonName="John Doe" OrganizationName=Clavister
     OrganizationalUnit=Support Country=Sweden
     EmailAddress=john.doe@Clavister.com


Device:/MyIDList> cc
```

Finally, apply the Identification List to the IPsec tunnel:

```
Device:/> set Interface IPsecTunnel MyIPsecTunnel AuthMethod=Certificate
     IDList=MyIDList RootCertificates=AdminCert GatewayCertificate=AdminCert
```

Web Interface

First create an Identification List:

1. Go to **Objects > VPN Objects > ID List > Add > ID List**
2. Enter a name for the list, for example *MyIDList*
3. Click **OK**

Then, create an ID:

1. Go to **Objects > VPN Objects > IKE ID List > Add > ID List**
2. In the grid control, click on **MyIDList**

3. Enter a name for the ID, for example *JohnDoe*
4. Select **Distinguished name** in the **Type** control
5. Now enter:
   - **Common Name:** John Doe
   - **Organization Name:**Clavister
   - **Organizational Unit:** Support
   - **Country:** Sweden
   - **Email Address:** john.doe@Clavister.com
6. Click **OK**

Finally, apply the Identification List to the IPsec tunnel:

1. Go to **Interfaces > IPsec**
2. In the grid control, click on the IPsec tunnel object of interest
3. Under the **Authentication** tab, choose **X.509 Certificate**
4. Select the appropriate certificate in the **Root Certificate(s)** and **Gateway Certificate** controls.
5. Select **MyIDList** in the **Identification List**.
6. Click **OK**

10.4. IPsec Tunnels

10.4.1. Overview

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 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 rule set 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 rule set. 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 rule set is consulted. Encrypted traffic to the gateway therefore does not need to be allowed in the rule set. This behavior can be changed in the ***IPsec advanced settings*** *section.*

IPsec Tunnel Quick Start

This section covers IPsec tunnels in some detail. A quick start checklist of setup steps for these protocols in typical scenarios can be found in the following sections:

- Section 10.2.1, “IPsec LAN to LAN with Pre-shared Keys”
- Section 10.2.2, “IPsec LAN to LAN with Certificates”
- Section 10.2.3, “IPsec Roaming Clients with Pre-shared Keys”
- Section 10.2.4, “IPsec Roaming Clients with Certificates”

10.4.2. LAN to LAN Tunnels with Pre-shared Keys

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 implementer 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:

- Set up the **VPN tunnel properties** and include the Pre-Shared key.
- Set up the **VPN tunnel properties**.
- Set up the **Route** in the *main* routing table (or another table if an alternate is being used).
- Set up the **Rules** (a 2-way tunnel requires 2 rules).

10.4.3. 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 its 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, irrespective 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 algorithm proposal lists that are pre-configured in CorePlus.

10.4.3.1. PSK based client tunnels

Example 10.4. 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.

Web Interface

A. Create a pre-shared key for IPsec authentication:

1. Go to **Objects > Authentication Objects > Add > Pre-Shared Key**
2. Now enter:
   - **Name:** Enter a name for the key, for example *SecretKey*
   - **Shared Secret:** Enter a secret passphrase
   - **Confirm Secret:** Enter the secret passphrase again
3. Click **OK**

B. Configure the IPsec tunnel:

1. Go to **Interfaces > IPsec > Add > IPsec Tunnel**
2. Now enter:
   - **Name:** RoamingIPsecTunnel
   - **Local Network:** 10.0.1.0/24 (This is the local network that the roaming users will connect to)

- **Remote Network:** all-nets
- **Remote Endpoint:** (None)
- **Encapsulation Mode:** Tunnel

3. For Algorithms enter:
   - **IKE Algorithms:** Medium or High
   - **IPsec Algorithms:** Medium or High
4. For Authentication enter:
   - **Pre-Shared Key:** Select the pre-shared key created earlier
5. Under the **Routing** tab:
   - Enable the option: **Dynamically add route to the remote network when a tunnel is established.**
6. Click **OK**

C. Finally configure the IP rule set to allow traffic inside the tunnel.

10.4.3.2. Self-signed Certificate based client tunnels

Example 10.5. 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.

Web Interface

A. Create a Self-signed Certificate for IPsec authentication:

The step to actually create self-signed certificates is performed outside the WebUI using a suitable software product. The certificate should be in the PEM (Privacy Enhanced Mail) file format.

B. Upload all the client self-signed certificates:

1. Go to **Objects > Authentication Objects > Add > Certificate**
2. Enter a suitable name for the Certificate object.
3. Select the **X.509 Certificate** option
4. Click **OK**

C. Create Identification Lists:

1. Go to **Objects > VPN Objects > ID List > Add > ID List**
2. Enter a suitable **name**, for example *sales*
3. Click **OK**
4. Go to **Objects > VPN Objects > ID List > Sales > Add > ID**
5. Enter the **name** for the client
6. Select **Email** as **Type**
7. In the **Email address** field, enter the email address selected when you created the certificate on the client
8. 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 > Add > IPsec Tunnel**
2. Now enter:
   - **Name:** RoamingIPsecTunnel
   - **Local Network:** 10.0.1.0/24 (This is the local network that the roaming users will connect to)
   - **Remote Network:** all-nets
   - **Remote Endpoint:** (None)
   - **Encapsulation Mode:** Tunnel
3. For Algorithms enter:
   - **IKE Algorithms:** Medium or High
   - **IPsec Algorithms:** Medium or High
4. For Authentication enter:
   - Choose X.509 Certificate as authentication method
   - **Root Certificate(s):** Select all your client certificates and add them to the **Selected** list
   - **Gateway Certificate:** Choose your newly created gateway certificate
   - **Identification List:** Select your ID List that you want to associate with your VPN Tunnel. In our case that will be **sales**
5. Under the **Routing** tab:
   - Enable the option: **Dynamically add route to the remote network when a tunnel is established.**
6. Click **OK**

E. Finally configure the IP rule set to allow traffic inside the tunnel.

10.4.3.3. CA Server issued Certificates based client tunnels

Setting up client tunnels using a CA issued 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 4.7, “Certificates”.

It is the responsibility of the administrator to acquire the appropriate certificate from an issuing authority for client tunnels. 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 4.7, “Certificates”.

Example 10.6. 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.

Web Interface

A. Upload all the client certificates:

1. Go to **Objects > Authentication Objects > Add > Certificate**
2. Enter a suitable name for the Certificate object.

3. Select the **X.509 Certificate** option
4. Click **OK**

B. Create Identification Lists:

1. Go to **Objects > VPN Objects > ID List > Add > ID List**
2. Enter a descriptive **name**, for example *sales*
3. Click **OK**
4. Go to **Objects > VPN Objects > ID List > Sales > Add > ID**
5. Enter the **name** for the client
6. Select **Email** as **Type**
7. In the **Email address** field, enter the email address selected when you created the certificate on the client
8. Create a new ID for every client that you want to grant access rights according to the instructions above

C. Configure the IPsec tunnel:

1. Go to **Interfaces > IPsec > Add > IPsec Tunnel**
2. Now enter:
   - **Name:** RoamingIPsecTunnel
   - **Local Network:** 10.0.1.0/24 (This is the local network that the roaming users will connect to)
   - **Remote Network:** all-nets
   - **Remote Endpoint:** (None)
   - **Encapsulation Mode:** Tunnel
3. For Algorithms enter:
   - **IKE Algorithms:** Medium or High
   - **IPsec Algorithms:** Medium or High
4. For Authentication enter:
   - Choose **X.509 Certificates** as the authentication method
   - **Root Certificate(s):** Select your CA server root certificate imported earlier and add it to the **Selected** list
   - **Gateway Certificate:** Choose your newly created gateway certificate
   - **Identification List:** Select your ID List that you want to associate with your VPN Tunnel. In our case that will be **sales**
5. Under the **Routing** tab:
   - Enable the option: **Dynamically add route to the remote network when a tunnel is established**
6. Click **OK**

D. Finally configure the IP rule set to allow traffic inside the tunnel.

10.4.3.4. Using Config Mode

IKE Configuration Mode (Config Mode) is an extension to IKE that allows CorePlus to provide LAN configuration information to remote VPN clients. It is used to dynamically configure IPsec clients with IP addresses and corresponding netmasks, and to exchange other types of information associated with DHCP. The IP address provided to a client can be either be based on a range of predefined static IP addresses defined for Config Mode or it can come from DHCP servers associated with an *IP Pool* object.

An IP pool is a cache of IP addresses collected from DHCP servers and leases on these addresses are automatically renewed when the lease time is about to expire. IP Pools also manage additional information such as DNS and WINS/NBNS, just as an ordinary DHCP server would. (For detailed information on pools see Section 6.5, “IP Pools”.)

Defining the Config Mode Object

Currently only one Config Mode object can be defined in CorePlus and this is referred to as the *Config Mode Pool* object. The key parameters associated with it are as follows:

Use Pre-defined IP Pool Object	The IP Pool object that provides the IP addresses.
Use a Static Pool	As an alternative to using an IP Pool, a static set of IP addresses can be defined.
DNS	The IP address of the DNS used for URL resolution (already provided by an IP Pool).
NBNS/WINS	The IP address for NBNS/WINS resolution (already provided by an IP Pool).
DHCP	Instructs the host to send any internal DHCP requests to this address.
Subnets	A list of the subnets that the client can access.

Example 10.7. Setting Up Config Mode

In this example the Config Mode Pool object is enabled by associating with it an already configured IP Pool object called *ip_pool1*

Web Interface

1. Go to **Objects > VPN Objects > IKE Config Mode Pool**
2. The Config Mode Pool object properties web page now appears
3. Select **Use a pre-defined IPPool object**
4. Choose the *ip_pool1* object from the **IP Pool** drop-down list
5. Click **OK**

After defining the Config Mode object, the only remaining action is to enable Config Mode to be used with the IPsec Tunnel.

Example 10.8. Using Config Mode with IPsec Tunnels

Assuming a predefined tunnel called *vpn_tunnel1* this example shows how to enable Config Mode for that tunnel.

Web Interface

- Go to **Interfaces > IPsec**
- Select the tunnel *vpn_tunnel1* for editing
- Select **IKE Config Mode** drop down list
- Click **OK**

IP Validation

CorePlus always checks if the source IP address of each packet inside an IPsec tunnel is the same as the IP address assigned to the IPsec client with IKE Config Mode. If a mismatch is detected the packet is always dropped and a log message generated with a severity level of **Warning**. This message includes the two IP addresses as well as the client identity.

Optionally, the affected SA can be automatically deleted if validation fails by enabling the advanced setting **IPsecDeleteSAOnIPValidationFailure**. The default value for this setting is *Disabled*.

10.4.4. Fetching CRLs from an alternate LDAP server

A Root Certificate usually includes the IP address or hostname of the Certificate Authority to contact when certificates or CRLs 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 10.9. Setting up an LDAP server

This example shows how to manually setup and specify a LDAP server.

CLI

```
Device:/> add LDAPServer Host=192.168.101.146 Username=myusername
     Password=mypassword Port=389
```

Web Interface

1. Go to **Objects > VPN Objects > LDAP > Add > LDAP Server**
2. Now enter:
   - **IP Address:** 192.168.101.146
   - **Username:** myusername
   - **Password:** mypassword
   - **Confirm Password:** mypassword
   - **Port:** 389
3. Click **OK**

10.4.5. Troubleshooting with *ikesnoop*

VPN Tunnel Negotiation

When setting up IPsec tunnels, problems can arise because the initial negotiation fails when the devices at either end of a VPN tunnel try but fail to agree on which protocols and encryption methods will be used. 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 *ikesnoop* command can be entered via a CLI console or directly via the RS232 Console.

To begin monitoring the full command is:

```
Device:/> ikesnoop -on -verbose
```

This means that ikesnoop output will be sent to the console for every VPN tunnel IKE negotiation. The output can be overwhelming so to limit the output to a single IP address, for example the IP address *10.1.1.10*, the command would be:

```
Device:/> ikesnoop -on 10.1.1.10 -verbose
```

The IP address used is the IP address of the VPN tunnel's remote endpoint (either the IP of the remote endpoint or the client IP). To turn off monitoring, the command is:

```
Device:/> ikesnoop -off
```

The output from *verbose* option can be troublesome to interpret by an administrator seeing it for the first time. Presented below is some typical *ikesnoop* output with annotations to explain it. The 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 the *CLI Reference Guide*.

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 negotiation and the *server* refers to the device which is the *responder*.

Step 1. Client Initiates Exchange by Sending a Supported Algorithm List

The *verbose* option output initially shows the proposed list of algorithms that the client first sends to the server. This list details the protocols and encryption methods it can support. The purpose of the algorithm list is that the client is trying to find a matching set of protocols/methods supported by the server. The server examines the list and attempts to find a combination of the protocols/methods sent by the client which it can support. This matching process is one of the key purposes of the IKE exchange.

```
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, e.g. 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 a List of Supported IPsec Algorithms

Now the client sends the list of supported IPsec algorithms to the server. It will also contain the proposed host/networks that are 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 client's 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 a List of Supported Algorithms

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
```

10.4.6. IPsec Advanced Settings

The following CorePlus advanced settings are available for configuring IPsec tunnels.

IPsec Max Rules

This specifies the total number of IP rules that can be connected to IPsec tunnels. By default this is initially approximately 4 times the licensed **IPsecMaxTunnels** and system memory for this is allocated at startup. By reducing the number of rules, memory requirements can be reduced but making this change is not recommended.

IPsec Max Rules will always be reset automatically to be approximately 4 times **IPsec Max Tunnels** if the latter is changed. This linkage is broken once **IPsec Max Rules** is altered manually so that subsequent changes to **IPsec Max Tunnels** will not cause an automatic change in **IPsec Max Rules**.

Default: *4 times the license limit of IPsec Max Tunnels*

IPsec Max Tunnels

Specifies the total number of tunnels allowed by CorePlus. This value is usually taken from the license but in situations where it is desirable to have less than the license value it can be reduced. System memory for the tunnels is allocated at startup and reducing this value can therefore reduce memory requirements.

A warning log message is generated automatically when 90% of this value is reached.

Default: *According to the licensed limit*

IKE Send Initial Contact

Determines whether or not IKE should send the "Initial Contact" notification message. This message is sent to each remote endpoint when a connection is opened to it and there are no previous IPsec SA using that gateway.

Default: *Enabled*

IKE Send CRLs

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.

Note that this setting requires a restart to take effect.

Default: *Enabled*

IPsec Before Rules

Pass IKE & IPsec (ESP/AH) traffic sent to CorePlus directly to the IPsec engine without consulting the rule set.

Default: *Enabled*

IKE CRL Validity Time

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 IKECRLVailityTime expires or when the "next update" time occurs. Whichever happens first.

Default: *86400*

IKE Max CA Path

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 as "trusted" is found, or until it is determined that none of the certificates are trusted.

If there are more certificates in this path than what this setting specifies, the user certificate will be considered invalid.

Default: *15*

IPsec Cert Cache Max Certs

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*

IPsec Gateway Name Cache Time

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*

DPD Metric

The amount of time in tens of seconds that the peer is considered to be alive (reachable) since the last received IKE message. This means that no DPD messages for checking aliveness of the peer will be sent during this time even though no packets from the peer have been received during this time.

In other words, the amount of time in tens of seconds that a tunnel is without traffic or any other sign of life before the peer is considered dead. If DPD is due to be triggered but other evidence of life is seen (such as IKE packets from the other side of the tunnel) within the time frame, no messages will be sent.

For example, if the other side of the tunnel has not sent any ESP packets for a long period but at least one IKE-packet has been seen within the last *(10 x the configured value)* seconds, then CorePlus will not send more messages to the other side.

Default: *3* (in other words, 3 x 10 = 30 seconds)

DPD Keep Time

The amount of time in tens of seconds that a peer is assumed to be dead after CorePlus has detected it to be so. While the peer is considered dead, CorePlus will not try to re-negotiate the tunnel or send DPD messages to the peer. However, the peer will not be considered dead any more as soon as a packet from it is received.

A more detailed explanation for this setting is that it is the amount of time in tens of seconds that an SA wil remain in the dead cache after a delete. An SA is put in the dead cache when the other side of the tunnel has not responded to messages for *DPD Expire Time* x 10 seconds and there is no other evidence of life. When the SA is placed in the dead cache, CorePlus will not try to re-negotiate the tunnel. If traffic that is associated with the SA that is in the dead cache is received, the SA will be removed from the dead cache. DPD will not trigger if the SA is already cached as dead.

This setting is used with IKEv1 only.

Default: *2* (in other words, 2 x 10 = 20 seconds)

DPD Expire Time

The length of time in seconds for which DPD messages will be sent to the peer. If the peer has not responded to messages during this time it is considered to be dead (in other words, un-reachable).

In other words, the length of time in seconds for which messages will be sent. If the other side of the tunnel has not sent a response to any messages then it is considered to be dead (not reachable). The SA will then be placed in the dead cache.

This setting is used with IKEv1 only.

Default: *15 seconds*

10.5. 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. The most commonly used feature that is relevant in this scenario is the ability of CorePlus to act as either a PPTP or L2TP server and the first two sections below deal with this. The third section deals with the further ability of CorePlus to act as a PPTP or L2TP client.

PPTP/L2TP Quick Start

This section covers L2TP and PPTP in some detail. A quick start checklist of setup steps for these protocols in typical scenarios can be found in the following sections:

- Section 10.2.5, “L2TP Roaming Clients with Pre-Shared Keys”
- Section 10.2.6, “L2TP Roaming Clients with Certificates”
- Section 10.2.7, “PPTP Roaming Clients”

10.5.1. PPTP Servers

Overview

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 D, *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.

Implementation

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 - IP protocol 47). 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 (TCP port 1723 is used). 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.

Deployment

PPTP offers a convenient solution to client access that is simple to deploy. PPTP does not 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 does not 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.

Troubleshooting PPTP

A common problem with setting up PPTP is that a router and/or switch in a network is blocking

TCP port 1723 and/or IP protocol 47 before the PPTP connection can be made to the Clavister Security Gateway. Examining the log can indicate if this problem occurred, with a log message of the following form appearing:

```
Error PPP lcp_negotiation_stalled ppp_terminated
```

Example 10.10. 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.

CLI

```
Device:/> add Interface L2TPServer MyPPTPServer ServerIP=ip_lan Interface=any
     IP=ip_wan IPPool=pp2p_Pool TunnelProtocol=PPTP AllowedRoutes=all-nets
```

Web Interface

1. Go to **Interfaces > PPTP/L2TP Servers > Add > PPTP/L2TP Server**
2. Enter a name for the PPTP Server, for example *MyPPTPServer*
3. Now enter:
   - **Inner IP Address:** ip_lan
   - **Tunnel Protocol:** PPTP
   - **Outer Interface Filter:** any
   - **Outer Server IP:** ip_wan
4. Under the **PPP Parameters** tab, select **pptp_Pool** in the **IP Pool** control
5. Under the **Add Route** tab, select **all_nets** from **Allowed Networks**
6. 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 will not be covered in this example.

10.5.2. L2TP Servers

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 10.11. Setting up an L2TP server

This example shows how to setup a L2TP Network Server. The example assumes 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.

CLI

```
Device:/> add Interface L2TPServer MyL2TPServer ServerIP=ip_l2tp
     Interface=l2tp_ipsec IP=ip_wan IPPool=L2TP_Pool TunnelProtocol=L2TP
     AllowedRoutes=all-nets
```

Web Interface

1. Go to **Interfaces > L2TP Servers > Add > L2TPServer**
2. Enter a suitable name for the L2TP Server, for example *MyL2TPServer*
3. Now enter:
   - **Inner IP Address:** ip_l2tp
   - **Tunnel Protocol:** L2TP
   - **Outer Interface Filter:** l2tp_ipsec
   - **Outer Server IP:** ip_wan
4. Under the **PPP Parameters** tab, select **L2TP_Pool** in the **IP Pool** control
5. Under the **Add Route** tab, select **all_nets** in the **Allowed Networks** control
6. 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 10.12. Setting up an L2TP Tunnel Over IPsec

This example shows how to setup a fully working L2TP Tunnel based on IPsec encryption and will cover many parts of basic VPN configuration. Before starting, you need to configure some address objects, for example 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.

A. Start by preparing a new Local User Database:

CLI

```
Device:/> add LocalUserDatabase UserDB
```

```
Device:/> cc LocalUserDatabase UserDB
```

```
Device:/UserDB> add User testuser Password=mypassword
```

Web Interface

1. Go to **User Authentication > Local User Databases > Add > Local User Database**
2. Enter a suitable for the user database, for instance UserDB
3. Go to **User Authentication > Local User Databases > UserDB > Add > User**
4. Now enter:
   - **Username:** testuser

- **Password:** mypassword
- **Confirm Password:** mypassword

5. 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.

B. Continue setting up the IPsec Tunnel:

CLI

```
Device:/> add Interface IPsecTunnel l2tp_ipsec LocalNetwork=ip_wan
     RemoteNetwork=all-nets IKEAlgorithms=Medium
     IPsecAlgorithms=esp-l2tptunnel PSK=MyPSK EncapsulationMode=Transport
     DHCPOverIPsec=Yes AddRouteToRemoteNet=Yes IPsecLifeTimeKilobytes=250000
     IPsecLifeTimeSeconds=3600
```

Web Interface

1. Go to **Interfaces > IPsec > Add > IPsec Tunnel**
2. Enter a name for the IPsec tunnel, for example *l2tp_ipsec*
3. Now enter:
   a. **Local Network:** ip_wan
   b. **Remote Network:** all-nets
   c. **Remote Endpoint:** none
   d. **Encapsulation Mode:** Transport
   e. **IKE Algorithms:** High
   f. **IPsec Algorithms:** esp-l2tptunnel
4. Enter 3600 in the **IPsec Life Time seconds** control
5. Enter 250000 in the **IPsec Life Time kilobytes** control
6. Under the **Authentication** tab, select **Pre-shared Key**
7. Select **MyPSK** in the **Pre-shared Key** control
8. Under the **Routing** tab, check the following controls:
   - **Allow DHCP over IPsec from single-host clients**
   - **Dynamically add route to the remote network when a tunnel is established**
9. 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.

C. Setup the L2TP Tunnel:

CLI

```
Device:/> add Interface L2TPServer l2tp_tunnel IP=ip_lan Interface=l2tp_ipsec
     ServerIP=ip_wan IPPool=l2tp_pool TunnelProtocol=L2TP
     AllowedRoutes=all-nets ProxyARPInterfaces=lan
```

Web Interface

1. Go to **Interfaces > L2TP Servers > Add > L2TPServer**

2. Enter a name for the L2TP tunnel, for example *l2tp_tunnel*
3. Now enter:
   - **Inner IP Address:** ip_lan
   - **Tunnel Protocol:** L2TP
   - **Outer Interface Filter:** l2tp_ipsec
   - **Server IP:** ip_wan
4. Under the **PPP Parameters** tab, check the **Use User Authentication Rules** control
5. Select **l2tp_pool** in the **IP Pool** control
6. Under the **Add Route** tab, select **all-nets** in the **Allowed Networks** control.
7. In the **ProxyARP** control, select the **lan** interface.
8. Click **OK**

In order to authenticate the users using the L2TP tunnel, a user authentication rule needs to be configured.

D. Next will be setting up the authentication rules:

CLI

```
Device:/> add UserAuthRule AuthSource=Local Interface=l2tp_tunnel
     OriginatorIP=all-nets LocalUserDB=UserDB agent=PPP TerminatorIP=ip_wan
     name=L2TP_Auth
```

Web Interface

1. Go to **User Authentication > User Authentication Rules > Add > UserAuthRule**
2. Enter a suitable name for the rule, for example *L2TP_Auth*
3. Now enter:
   - **Agent:** PPP
   - **Authentication Source:** Local
   - **Interface:** l2tp_tunnel
   - **Originator IP:** all-nets
   - **Terminator IP:** ip_wan
4. Under the **Authentication Options** tab enter *UserDB* as the **Local User DB**
5. Click **OK**

When the other parts are done, all that is left is the rules. To let traffic through from the tunnel, two IP rules should be added.

E. Finally, set up the rules:

CLI

First, change the current category to be the *main* IP rule set:

```
Device:/> cc IPRuleSet main
```

Now, add the IP rules:

```
Device:/main> add IPRule action=Allow Service=all_services
                SourceInterface=l2tp_tunnel SourceNetwork=l2tp_pool
                DestinationInterface=any DestinationNetwork=all-nets name=AllowL2TP


Device:/main> add IPRule action=NAT Service=all_services
                SourceInterface=l2tp_tunnel SourceNetwork=l2tp_pool
```

```
DestinationInterface=any DestinationNetwork=all-nets name=NATL2TP
```

Web Interface

1. Go to **Rules > IP Rules > Add > IPRule**
2. Enter a name for the rule, for example *AllowL2TP*
3. Now enter:
   - **Action:** Allow
   - **Service:** all_services
   - **Source Interface:** l2tp_tunnel
   - **Source Network:** l2tp_pool
   - **Destination Interface:** any
   - **Destination Network:** all-nets
4. Click **OK**
5. Go to **Rules > IP Rules > Add > IPRule**
6. Enter a name for the rule, for example *NATL2TP*
7. Now enter:
   - **Action:** NAT
   - **Service:** all_services
   - **Source Interface:** l2tp_tunnel
   - **Source Network:** l2tp_pool
   - **Destination Interface:** any
   - **Destination Network:** all-nets
8. Click **OK**

10.5.3. L2TP/PPTP Server advanced settings

The following L2TP/PPTP server advanced settings are available to the administrator:

L2TP Before Rules

Pass L2TP traffic sent to the Clavister Security Gateway directly to the L2TP Server without consulting the rule set.

Default: *Enabled*

PPTP Before Rules

Pass PPTP traffic sent to the Clavister Security Gateway directly to the PPTP Server without consulting the rule set.

Default: *Enabled*

Max PPP Resends

The maximum number of PPP layer resends.

Default: *10*

10.5.4. PPTP/L2TP Clients

The PPTP and L2TP protocols are described in the previous section. In addition to being able to act as a PPTP or L2TP server, CorePlus also offers the ability to act as a PPTP or L2TP clients. This can be useful if PPTP or L2TP is preferred as the VPN protocol instead of IPsec. One Clavister Security Gateway can act as a client and connect to another unit which acts as the server.

Client Setup

PPTP and L2TP shares a common approach to client setup which involves the following settings:

General Parameters

- **Name** - A symbolic name for the client
- **Interface Type** - Specifies if it is a PPTP or L2TP client.
- **Remote Endpoint** - The IP address of the remote endpoint. Where this is specified as a URL, the prefix *dns:* must be precede it.

Names of Assigned Addresses

Both PPTP and L2TP utilizes dynamic IP configuration using the *PPP LCP* protocol. When CorePlus receives this information, it is stored in symbolic host/network names. The settings for this are:

- **Inner IP Address** - The host name that is used for storing the assigned IP address. If this network object exists and has a value which is not *0.0.0.0* then the PPTP/L2TP client will try to get that one from the PPTP/L2TP server as the preferred IP.
- **Automatically pick name** - If this option is enabled then CorePlus will create a host name based on the name of the PPTP/L2TP interface, for example *ip_PPTPTunnel1*.
- **Primary/Secondary DNS Name** - This defines the DNS servers from a list of predefined network objects.

Note

A PPTP/L2TP server will not provide information such as gateway or broadcast addresses, as this is not used with PPTP/L2TP tunnels. When using PPTP/L2TP, the default route is normally routed directly across the PPTP/L2TP tunnel without a specified gateway.

Authentication

- **Username** - Specifies the username to use for this PPTP/L2TP interface.
- **Password** - Specifies the password for the interface.
- **Authentication** - Specifies which authentication protocol to use.
- **MPPE** - Specifies if *Microsoft Point-to-Point Encryption* is used and which level to use.

If **Dial On Demand** is enabled then the PPTP/L2TP tunnel will not be set up until traffic is sent on the interface. The parameters for this option are:

- **Activity Sense** - Specifies if dial-on-demand should trigger on **Send** or **Recv** or both.

- **Idle Timeout** - The time of inactivity in seconds to wait before disconnection.

Using the PPTP Client Feature

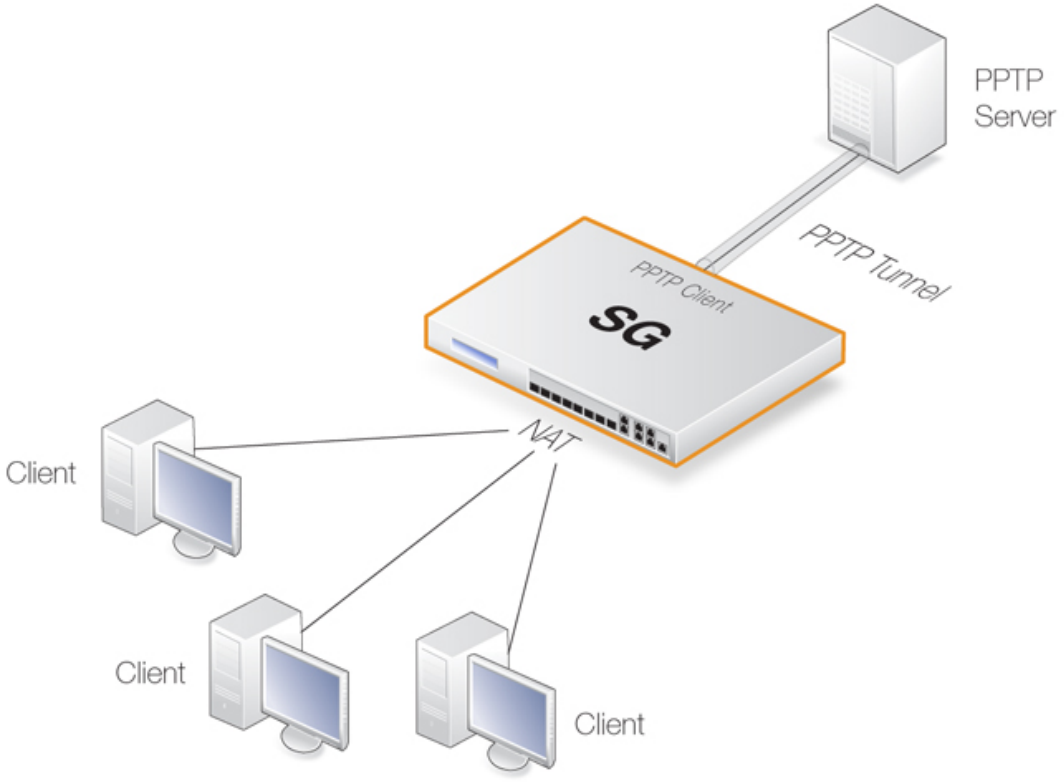
One usage of the PPTP client feature is shown in the scenario depicted below.

Here a number of clients are being NATed through CorePlus before being connected to a PPTP server on the other side of the Clavister Security Gateway. If more that one of the clients is acting as a PPTP client which is trying to connect to the PPTP server then this will not work because of the NATing.

The only way of achieving multiple PPTP clients being NATed like this, is for the Clavister Security Gateway to act as a PPTP client when it connects to the PPTP server. To summarize the setup:

- A PPTP tunnel is defined between CorePlus and the server.
- A route is added to the routing table in CorePlus which specifies that traffic for the server should be routed through the PPTP tunnel.

Figure 10.3. PPTP Client Usage



10.6. CA Server Access

Where certificates are used, the two sides of a VPN tunnel exchange their certificates during the tunnel setup negotiation and either may then try to validate the received certificate by accessing a *CA server*. A certificate contains a URL (the *CRL Distribution Point*) which specifies the validating CA server and server access is performed using an HTTP *PUT* request with an HTTP reply. (This URL is more correctly called an FQDN - *Fully Qualified Domain Name*.)

CA servers are of two types:

- A commercial CA server operated by one of the commercial certificate issuing companies. These are accessible over the public Internet and their FQDNs are resolvable through the public Internet DNS server system.
- A private CA server operated by the same organization setting up the VPN tunnels. The IP address of a private server will not be known to the public DNS system unless it is explicitly registered. It also will not be known to an internal network unless it is registered on an internal DNS server.

The following considerations should be taken into account for CA server access to succeed:

- Either side of a VPN tunnel may issue a validation request to a CA server.
- For a certificate validation request to be issued, the FQDN of the certificate's CA server must first be resolved into an IP address. The following scenarios are possible:

  1. The CA server is a private server behind the Clavister Security Gateway and the tunnels are set up over the public internet but to clients that will not try to validate the certificate sent by CorePlus

     In this case, the IP address of the private server needs only be registered on a private DNS server so the FQDN can be resolved. This private DNS server will also have to configured in CorePlus so it can be found when CorePlus issues a validation request. This will also be the procedure if the tunnels are being set up entirely internally without using the public internet.

  2. The CA server is a private server with tunnels set up over the public internet and with clients that will try to validate the certificate received from CorePlus. In this case the following must be done:

     a. A private DNS server must be configured so that CorePlus can locate the private CA server to validate the certificates coming from clients.

     b. The external IP address of the Clavister Security Gateway needs to be registered in the public DNS system so that the FQDN reference to the private CA server in certificates sent to clients can be resolved. For example, CorePlus may send a certificate to a client with an FQDN which is *ca.company.com* and this will need to be resolvable by the client to a public external IP address of the Clavister Security Gateway through the public DNS system.

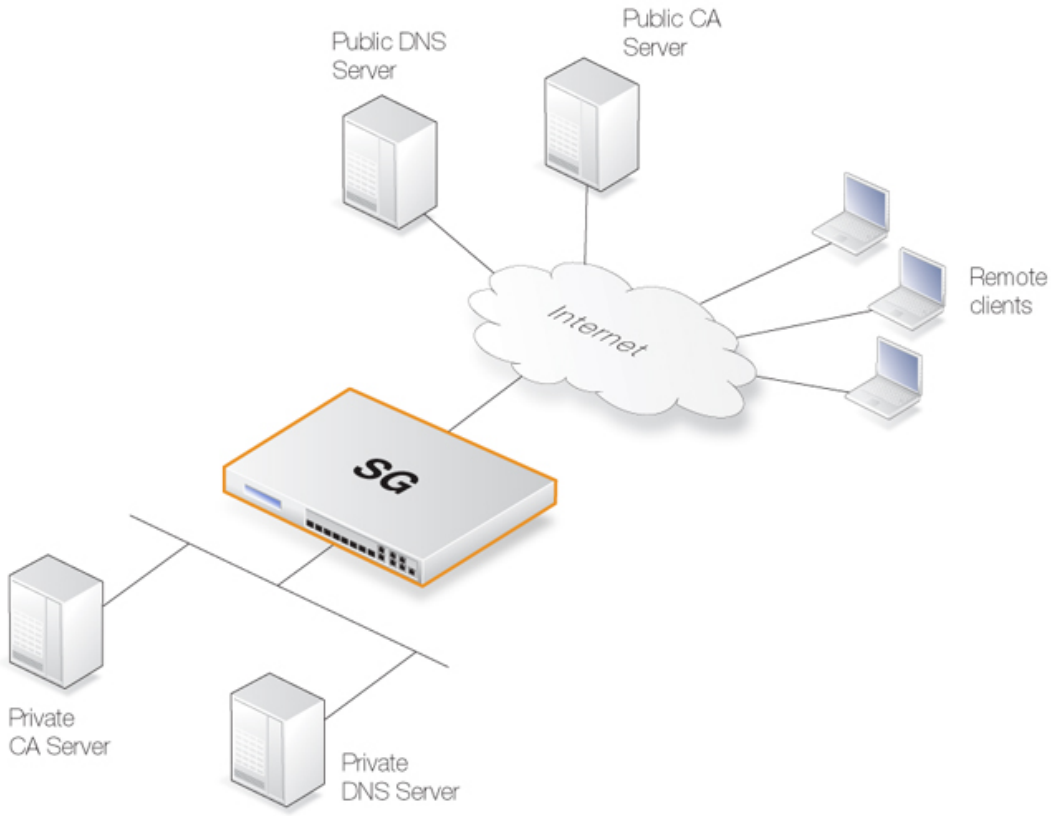
     The same steps should be followed if the other side of the tunnel is another security gateway instead of being many clients.

  3. The CA server is a commercial server on the public internet. In this, the simplest case, public DNS servers will resolve the FQDN. The only requirement is that CorePlus will need to have at least one public DNS server address configured to resolve the FQDNs in the certificates it receives.

- It must be also possible for an HTTP *PUT* request to pass from the validation request source (either the Clavister Security Gateway or a client) to the CA server and an HTTP reply to be received. If the request is going to pass through the Clavister Security Gateway, the appropriate

rules in the CorePlus IP rule set need to be defined to allow this traffic through.

Figure 10.4. Certificate Validation Components



CA Server Access by Clients

In a VPN tunnel with roaming clients connecting to the Clavister Security Gateway, the VPN client software may need to access the CA server. Not all VPN client software will need this access. In the Microsoft clients prior to Vista, CA server requests are not sent at all. With Microsoft Vista validation became the default with the option to disable it. Other non-Microsoft clients differ in the way they work but the majority will attempt to validate the certficate.

Placement of Private CA Servers

The easiest solution for placement of a private CA server is to have it on the unprotected side of the Clavister Security Gateway. This however, is not recommended from a security viewpoint. It is better to place it on the inside (or preferably in the DMZ if available) and to have CorePlus control access to it.

As explained previously, the address of the private CA server must be resolvable through public DNS servers for certificate validation requests coming from the public internet. If the certificate queries are coming only from the Clavister Security Gateway and the CA server is on the internal side of the security gateway then the IP address of the internal DNS server must be configured in CorePlus so that these requests can be resolved.

Turning Off FQDN Resolution

As explained in the troubleshooting section below, identifying problems with CA server access can be done by turning off the requirement to validate certificates. Attempts to access CA servers by CorePlus can be disabled with the **Disable CRLs** option for certificate objects. This means that checking against the CA server's revocation list will be turned off and access to the server will not be attempted.

10.7. VPN Troubleshooting

General Troubleshooting

In all types of VPNs some basic troubleshooting checks can be made:

- Check that all IP addresses have been specified correctly.
- Check that all pre-shared keys and usernames/passwords are correctly entered.
- Use ICMP *Ping* to confirm that the tunnel is working. With roaming clients this is best done by Pinging the internal IP address of the local network interface on the Clavister Security Gateway from a client (in LAN to LAN setups pinging could be done in any direction). If CorePlus is to able to respond to a Ping then the following rule must exist in the IP rule set.

Action	Src Interface	Src Network	Dest Interface	Dest Network	Service
Allow	vpn_tunnel	all-nets	core	all-nets	ICMP

- Ensure that another **IPsec Tunnel** definition is not preventing the correct definition being reached. The tunnel list is scanned from top to bottom by CorePlus and a tunnel in a higher position with the **Remote Network** set to *all-nets* and the **Remote Endpoint** set to *none* could prevent the correct tunnel being reached. A symptom of this is often an *Incorrect Pre-shared Key* message.
- Try and avoid duplication of IP addresses between the remote network being accessed by a client and the internal network to which a roaming client belongs.

  If a roaming client becomes temporarily part of a network such as a Wi-Fi network at an airport, the client will get an IP address from the Wi-Fi network's DHCP server. If that IP also belongs to the network behind the Clavister Security Gateway accessible through a tunnel, then Windows will still continue to assume that the IP address is to be found on the client's local network. Windows therefore will not correctly route packets bound for the remote network through the tunnel but instead route them to the local network.

  The solution to this problem of local/remote IP address duplication is to create a new route in the client's Windows routing table that explicitly routes the IP address to the tunnel.
- If roaming client user authentication is not asking the users for their username/password then ensure that the following advanced settings are enabled:
  - *IPsec Before Rules* for pure IPsec roaming clients.
  - *L2TP Before Rules* for L2TP roaming clients.
  - *PPTP Before Rules* for PPTP roaming clients.

  These settings should be enabled by default and they ensure that user authentication traffic between CorePlus and the client can bypass the IP rule set. If the appropriate setting is not enabled then an explicit rule needs to be added to the IP rule set to allow the authentication traffic to pass between roaming clients and CorePlus. This rule will have a destination interface of **core** (which means CorePlus itself).
- If the remote endpoint is specified as a URL, make sure that the URL string is preceded by the prefix *dns:*. If, for example, the tunnel remote endpoint is to be specified as *vpn.company.com*, this should be specified as *dns:vpn.company.com*.

Troubleshooting Certificates

If certificates have been used in a VPN solution then the following should be looked at:

- Check that the correct certificates have been used.
- Check that the certificate *.cer* and *.key* files have the same filename. For example, *my_cert.key* and *my_cert.cer*.
- Check that the certificates have not expired.
- Check that the CorePlus date and time is set correctly and consider time-zone issues with newly generated certificates (the time of generation may not be the same as the CA server's system time).
- Disable CRL (revocation list) checking to see if CA server access could be the problem. CA server. CA Server issues are discussed further in Section 10.6, “CA Server Access”.

Troubleshooting IPsec Tunnels

A number of commands can be used to diagnose IPsec tunnels:

The *ipsecstat* console command

ipsecstat can be used to show that IPsec tunnels have correctly established. A representative example of output is:

```
> ipsecstat

--- IPsec SAs:

Displaying one line per SA-bundle

IPsec Tunnel    Local Net         Remote Net       Remote GW
------------    --------------    ------------     -------------
L2TP_IPSec      214.237.225.43    84.13.193.179    84.13.193.179
IPsec_Tun1      192.168.0.0/24    172.16.1.0/24    82.242.91.203
```

To examine the first IKE negotiation phase of tunnel setup use:

```
> ipsecstat -ike
```

To get complete details of tunnel setup use:

```
> ipsecstat -u -v
```

The *ikesnoop* console command

A common problem with setting up IPsec is a list of proposed algirthms that is unacceptable to the device at the other end of the tunnel. The *ikesnoop* command is a useful tool for diagnosing incompatible algorithm proposal lists by showing the details of negotiations during tunnel setup. The basic form of this command is:

```
ikesnoop -on -verbose
```

Once issued, an ICMP *ping* can be then sent to the Clavister Security Gateway from the remote end of the tunnel. This will cause *ikesnoop* to output details of the tunnel setup negotiation to the console and any algorithm proposal list incompatibilities can be seen.

If there are multiple tunnels in a setup or multiple clients on a single tunnel then the output from *verbose* option can be overwhelming. It is therefore better to specify that the output comes from a

single tunnel by specifying the IP address of the tunnel's endpoint (this is either the IP of the remote endpoint or a client's IP address). The command takes the form:

```
ikesnoop -on <ip-address> -verbose
```

Ikesnoop can be turned off with the command:

```
ikesnoop -off
```

For a more detailed discussion of this topic, see Section 10.4.5, “Troubleshooting with *ikesnoop*”.

Management Interface Failure with VPN

If any VPN tunnel is set up and then the management interface no longer operates then it is likely to be a problem with the management traffic being routed back through the VPN tunnel instead of the correct interface.

This happens when a route is established in the main routing table which routes any traffic for **all-nets** through the VPN tunnel. If the management interface is not reached by the VPN tunnel then the administrator needs to create a specific route that routes management interface traffic leaving the Clavister Security Gateway back to the management subnet.

Chapter 11. Traffic Management

This chapter describes how CorePlus can manage network traffic.

11.1. Traffic Shaping

11.1.1. Introduction

QoS with TCP/IP

A weakness of TCP/IP is the lack of true *Quality of Service* (QoS) functionality. QoS is the ability to guarantee and limit network bandwidth for certain services and users. Solutions such as the *Differentiated Services* (Diffserv) architecture have been designed to try and deal with the QoS issue in large networks by using information in packet headers to provide network devices with QoS information.

CorePlus Diffserv Support

CorePlus supports the Diffserv architecture the following ways:

- CorePlus forwards the 6 bits which make up the Diffserv *Differentiated Services Code Point* (DSCP) as well as copying these bits from the data traffic inside VPN tunnels to the encapsulating packets.
- As described later in this chapter, DSCP bits can be used by the CorePlus traffic shaping subsystem as a basis for prioritizing traffic passing through the Clavister Security Gateway.

The Traffic Shaping Solution

Architectures like Diffserv however, fall short if applications themselves supply the network with QoS information. In most networks it is rarely appropriate to let the applications, the users of the network, decide the priority of their own traffic. If the users cannot be relied upon then the network equipment must make the decisions concerning priorities and bandwidth allocation.

CorePlus provides QoS control by allowing the administrator to apply limits and guarantees to the network traffic passing through the Clavister Security Gateway. This approach is often referred to as *traffic shaping* and is well suited to managing bandwidth for LANs as well as to managing the bottlenecks that might be found in larger WANs. It can be applied to any traffic including that passing through VPN tunnels.

Traffic Shaping Objectives

Traffic shaping operates by measuring and queuing IP packets with respect to a number of configurable parameters. The objectives are:

- Applying bandwidth limits and queuing packets that exceed configured limits, then sending them later when bandwidth demands are lower.

- Dropping packets if packet buffers are full. The packets to be dropped should be chosen from those that are responsible for the "jam".
- Prioritizing traffic according to administrator decisions. If traffic with a high priority increases while a communications line is full, traffic with a low priority can be temporarily limited to make room for the higher priority traffic.
- Providing bandwidth guarantees. This is typically accomplished by treating a certain amount of traffic (the guaranteed amount) as high priority. Traffic exceeding the guarantee then has the same priority as "any other traffic", and competes with the rest of the non-prioritized traffic.

Traffic shaping does not typically work by queuing up immense amounts of data and then sorting out the prioritized traffic to send before sending non-prioritized traffic. Instead, the amount of prioritized traffic is measured and the non-prioritized traffic is limited dynamically so that it will not interfere with the throughput of prioritized traffic.

11.1.2. Traffic Shaping in CorePlus

CorePlus offers extensive traffic shaping capabilities for the packets passing through the Clavister Security Gateway. Different rate limits and traffic guarantees can be created as policies based on the traffic's source, destination and protocol, similar to the way in which IP rule set policies are created.

The two key components for traffic shaping in CorePlus are:

- **Pipes**
- **Pipe Rules**

Pipes

A Pipe is the fundamental object for traffic shaping and is a conceptual channel through which packets of data can flow. It has various characteristics that define how traffic passing through it is handled. As many pipes as are required can be defined by the administrator. None are defined by default.

Pipes are simplistic in that they do not care about the types of traffic that pass through them nor the direction of that traffic. They simply measure the data that passes through them and apply the administrator configured limits for the pipe as a whole or for *Precedences* and/or *Groups* (these are explained below).

CorePlus is capable of handling hundreds of pipes simultaneously, but in reality most scenarios require only a handful of pipes. It is possible dozens of pipes may be needed in scenarios where individual pipes are used for individual protocols (or in ISP cases, clients).

Pipe Rules

Pipe Rules make up the *Pipe Rule set*. Each Rule is defined much like other CorePlus policies: by specifying the source/destination interface/network as well as the Service to which the rule is to apply. Once a new connection is permitted by the IP rule set, the Pipe rule set is always checked for matching rules and in the same way, by going from top to bottom. The first matching Pipe Rule, if any, is used for traffic shaping. The Pipe rule set is initially empty.

When a Pipe Rule is defined, the pipes to be used with that rule are also specified and they are placed into one of two lists in the Pipe Rule. These lists are:

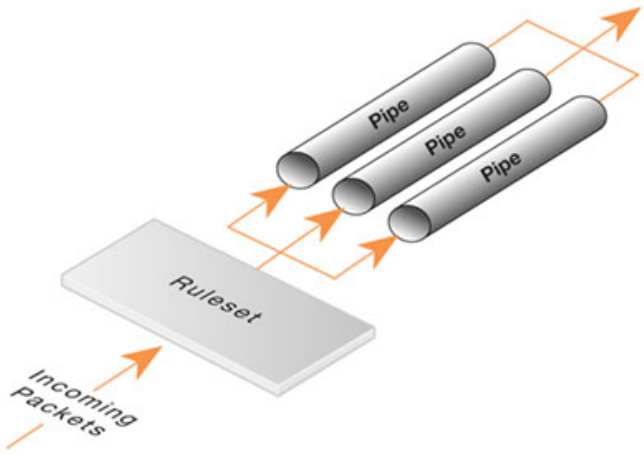
- **The Forward Chain**

  These are the pipes that will be used for outgoing (leaving) traffic from the Clavister Security Gateway. One, none or a series of pipes may be specified.

- **The Return Chain**

  These are the pipes that will be used for incoming (arriving) traffic. One, none or a series of pipes may be specified.

Figure 11.1. Pipe rule set to Pipe Packet Flow



Where one pipe is specified in a list then that is the pipe whose characteristics will be applied to the traffic. If a series of pipes are specified then these will form a *Chain* of pipes through which traffic will pass. A chain can be made up of at most 8 pipes.

If no pipe is specified in a list then traffic that matches the rule will not flow through any pipe but it will also mean that the traffic will not be subject to any other pipe rules found later in the rule set.

11.1.3. Simple Bandwidth Limiting

The simplest use of pipes is for bandwidth limiting. This is also a scenario that does not require much planning. The example that follows applies a bandwidth limit to inbound traffic only. This is the direction most likely to cause problems for Internet connections.

Example 11.1. 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.

CLI

```
Device:/> add Pipe std-in LimitKbpsTotal=2000
```

Web Interface

1. Go to **Traffic Management > Traffic Shaping > Pipes > Add > Pipe**
2. Specify a suitable name for the pipe, for instance *std-in*
3. Enter *2000* in the **Total** textbox under **Pipe Limits**
4. Click **OK**

Traffic needs to be passed through the pipe and this is done by using the pipe in a Pipe Rule.

We will use the above pipe to limit inbound traffic. This limit will apply to 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.

CLI

```
Device:/> add PipeRule ReturnChain=std-in SourceInterface=lan
     SourceNetwork=lannet DestinationInterface=wan
     DestinationNetwork=all-nets Service=all_services name=Outbound
```

Web Interface

1. Go to **Traffic Management > Traffic Shaping > Add > Pipe Rule**
2. Specify a suitable name for the pipe, for instance *outbound*.
3. Now enter:
   - **Service:** all_services
   - **Source Interface:** lan
   - **Source Network:** lannet
   - **Destination Interface:** wan
   - **Destination Network:** all-nets
4. Under the **Traffic Shaping** tab, make std-in selected in the **Return Chain** control.
5. Click **OK**

This setup limits all traffic from the outside (the Internet) to 2 megabits per second. No priorities are applied, nor any dynamic balancing.

11.1.4. Limiting Bandwidth in Both Directions

A single pipe does not care which direction the traffic through it is coming from when it calculates total throughout. Using the same pipe for both outbound and inbound traffic is allowed by CorePlus but it will not neatly partition pipe limits between the two directions. The following scenario clarifies this.

In the previous example only bandwidth in the inbound direction is limited. We chose this direction because in most setups, it is the direction that becomes full first. Now, what if we want to limit outbound bandwidth in the same way?

Just inserting **std-in** in the forward chain will not work since you probably want 2 Mbps limit for outbound traffic to be separate from the 2 Mbps limit for inbound traffic. If we try to pass 2 Mbps of outbound traffic through the pipe in addition to 2 Mbps of inbound traffic, it adds up to 4 Mbps. Since the pipe limit is 2 Mbps, you'd get something close to 1 Mbps in each direction.

Raising the total pipe limit to 4 Mbps will not solve the problem since the single pipe will not know that 2 Mbps inbound and 2 Mbps outbound was intended. 3 Mbps outbound and 1 Mbps inbound might be the result since that also adds up to 4 Mbps.

The recommended way to control bandwidth in both directions is to use two separate pipes one for inbound and one for outbound traffic. In the scenario under discussion each pipe would have a 2 Mbps limit to achieve the desired result. The following example goes through the setup for this.

Example 11.2. Limiting Bandwidth in Both Directions

Create a second pipe for outbound traffic:

CLI

```
Device:/> add Pipe std-out LimitKbpsTotal=2000
```

Web Interface

1. Go to **Traffic Management > Traffic Shaping > Pipes > Add > Pipe**
2. Specify a name for the pipe, for example *std-out*
3. Enter 2000 in **Total** textbox
4. Click **OK**

After creating a pipe for outbound bandwidth control, add it to the forward pipe chain of the rule created in the previous example:

CLI

```
Device:/> set PipeRule Outbound ForwardChain=std-out
```

Web Interface

1. Go to **Traffic Management > Traffic Shaping > Pipe Rules**
2. Right-click on the piperule you created in the previous example and choose **Edit**
3. Under the **Traffic Shaping** tab, select *std-out* in the **Forward Chain** list
4. Click **OK**

This results in all outbound connections being limited to 2 Mbps in each direction.

11.1.5. Creating Differentiated Limits with Chains

In the previous examples a static traffic limit for all outbound connections was applied. What if we want to limit web surfing more than other traffic? We could set up two "surfing" pipes for inbound and outbound traffic. However, we most likely will not need to limit outbound traffic because surfing usually consists of short outbound requests followed by long inbound answers. Let's assume the total bandwidth limit is 250 kbps and 125 kbps of that is to be allocated to web surfing inbound traffic. A **surf-in** pipe is therefore setup for inbound traffic with a 125 kbps limit.

Next a new Pipe Rule is set up for surfing that uses the **surf-in** pipe and it is placed before the rule that directs "everything else" through the **std-in** pipe. That way surfing traffic goes through the **surf-in** pipe and everything else is handled by the rule and pipe created earlier.

Unfortunately this will not achieve the desired effect, which is allocating a maximum of 125 kbps to inbound surfing traffic as part of the 250 kbps total. Inbound traffic will pass through one of two pipes: one that allows 250 kbps, and one that allows 125 kbps, giving a possible total of 375 kbps of inbound traffic.

To solve this we create a *chain* of the **surf-in** pipe followed by the **std-in** pipe in the surfing traffic Pipe Rule. Inbound surf traffic will now first pass through **surf-in** and be limited to a maximum of 125 kbps. Then, it will pass through the **std-in** pipe along with other inbound traffic, which will apply the 250 kbps total limit. If surfing uses the full limit of 125 kbps, those 125 kbps will occupy half of the **std-in pipe** leaving 125 kbps for the rest of the traffic. If no surfing is taking place then all of the 250 kbps allowed through **std-in** will be available for other traffic.

This is not a bandwidth guarantee for web browsing but it is a 125 kbps bandwidth guarantee for everything except web browsing. For web browsing the normal rules of first-come, first-forwarded will apply when competing for bandwidth. This may mean 125 kbps, but it may also mean much

slower speed if the connection is flooded.

Setting up pipes in this way only puts limits on the maximum values for certain traffic types. It does not give priorities to different types of competing traffic.

11.1.6. Precedences

All packets that pass through CorePlus traffic shaping pipes have a precedence. In the examples so far, precedences have not been explicitly set and so all packets have had the same default precedence of 0.

Eight precedences exist, numbered from 0 to 7. Precedence 0 is the least important and 7 is the most important. A precedence can be viewed as a separate traffic queue; traffic in precedence 2 will be forwarded before traffic in precedence 0, precedence 4 forwarded before 2.

The meaning of a precedence comes from the fact that it is either higher or lower than another precedence. If, for example, two precedences are used in a scenario, choosing 4 and 6 instead of 0 and 3 will makes no difference.

Figure 11.2. The Eight Pipe Precedences.

Allocating Precedence

The way precedence is assigned to a packet is decided by the Pipe Rule that controls it and is done in one of three ways:

- Use the precedence of the first pipe - Each pipe has a *default precedence* and packets take the default precedence of the first pipe they pass through.
- Use the allocated precedence - The Pipe Rule explicitly allocates a precedence.
- Use the DSCP bits - Take the precedence from the DSCP bits in the packet. DSCP is a subset of the Diffserv architecture where the *Type of Service* (ToS) bits are included in the IP packet header.

Pipe Precedences

When a pipe is configured, a *Default Precedence*, a *Minimum Precedence* and a *Maximum Precedence* can be specified. The Default Precedence is the precedence taken by a packet if it is not explicitly assigned by a Pipe Rule as described in the preceding paragraph.

The minimum and maximum precedences define the precedence range that the pipe will handle. If a packet arrives with an already allocated precedence below the minimum then its precedence is changed to the minimum. Similarly, if a packet arrives with an already allocated precedence above

the maximum, its precedence is changed to the maximum.

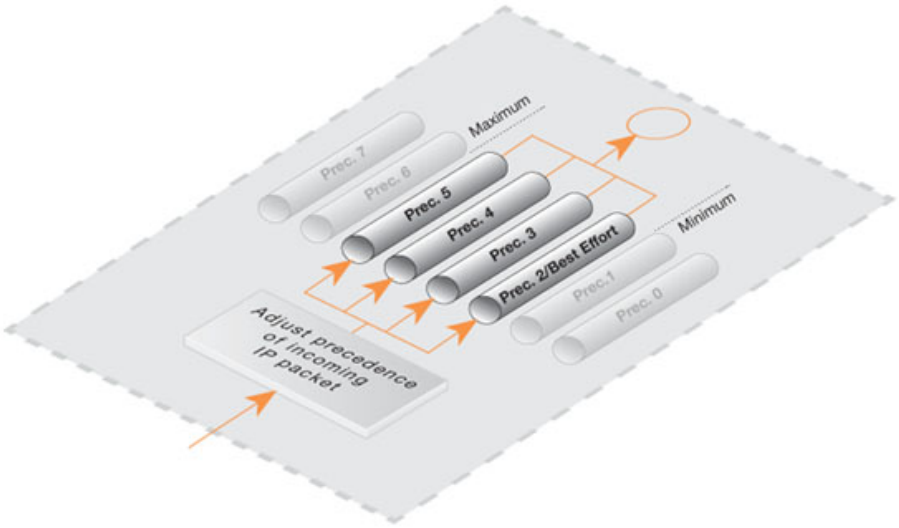
For each pipe, separate bandwidth limits may be optionally specified for each precedence level. These limits can be specified in kilobits per second and/or packets per second (if both are specified then the first limit reached will be the limit used). In precedences are used then the total limit for the pipe as a whole must be specified so the pipe knows when what its capacity is and therefore when precedences are used.

The *Best Effort* Precedence

The precedence defined as the minimum pipe precedence has a special meaning: it acts as the *Best Effort Precedence*. All packets arriving at this precedence will always be processed on a "first come, first forwarded" basis and cannot be sent to another precedence.

Packets with a higher precedence and that exceed the limits of that precedence will automatically be transferred down into this Best Effort precedence and they will no longer be treated differently from packets with lower priorities. This approach is used since a precedence limit is also a guarantees for that precedence.

Figure 11.3. Minimum and Maximum Pipe Precedence.



Precedences have no effect until the total bandwidth allocated for a pipe is reached. In other words when the pipe is "full". At that point traffic is prioritized by CorePlus with higher precedence packets being sent before lower precedence packets. The lower precedence packets are buffered. If buffer space becomes exhausted then they are dropped.

Applying Precedences

Continuing from the previous example, we add the requirement that SSH and Telnet traffic is to have a higher priority than all other traffic. To do this we add a Pipe Rule specifically for SSH and Telnet and set the priority in the rule to be a higher priority, say 2. We specify the same pipes in this new rule as are used for other traffic.

The effect of doing this is that the SSH and Telnet rule sets the higher priority on packets related to these services and these packets are sent through the same pipe as other traffic. The pipe then makes sure that these higher priority packets are sent first when the total bandwidth limit specified in the pipe's configuration is exceeded. Lower priority packets will be buffered and sent when higher priority traffic uses less than the maximum specified for the pipe. The buffering process is sometimes referred to as "throttling back" since it reduces the flow rate.

The Need for Guarantees

A problem can occur however if the prioritized traffic is a continuous stream such as real-time audio, resulting in continuous use all available bandwidth and resulting in unacceptably long queuing times for other services such as surfing, DNS or FTP. A means is therefore required to ensure that lower priority traffic gets some portion of bandwidth and this is done with *Bandwidth Guarantees*.

11.1.7. Guarantees

Bandwidth guarantees ensure that there is a minimum amount of bandwidth available for a given precedence. This is done by specifying a maximum limit for the precedence in a pipe. This will be the maximum amount of bandwidth that the precedence will accept and will send ahead of lower precedences. Excess traffic above this limit will be sent at the Best Effort precedence, behind traffic at precedences higher than Best Effort.

To change the prioritized SSH and Telnet traffic from the previous example to a 96 kbps guarantee, you set the precedence 2 limit for the *std-in*pipe to be 96 kbps.

This does not mean that inbound SSH and Telnet traffic is limited to 96 kbps. Limits in precedences above the Best Effort precedence will only limit how much of the traffic gets to pass in that specific precedence.

If more than 96 kbps of precedence 2 traffic arrives, any excess traffic will be moved down to the Best Effort precedence. All traffic at the Best Effort precedence is then forwarded on a first-come, first-forwarded basis.

Note
Setting a maximum limit for the lowest (Best Effort) precedence or any lower precedences has no meaning and will be ignored by CorePlus.

11.1.8. Differentiated Guarantees

A problem arises if 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, respectively.

Then, split the 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 rule set, you can easily change the precedence of all SSH and Telnet traffic by changing the default precedence of the **ssh-in** and **telnet-in** pipes.

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.

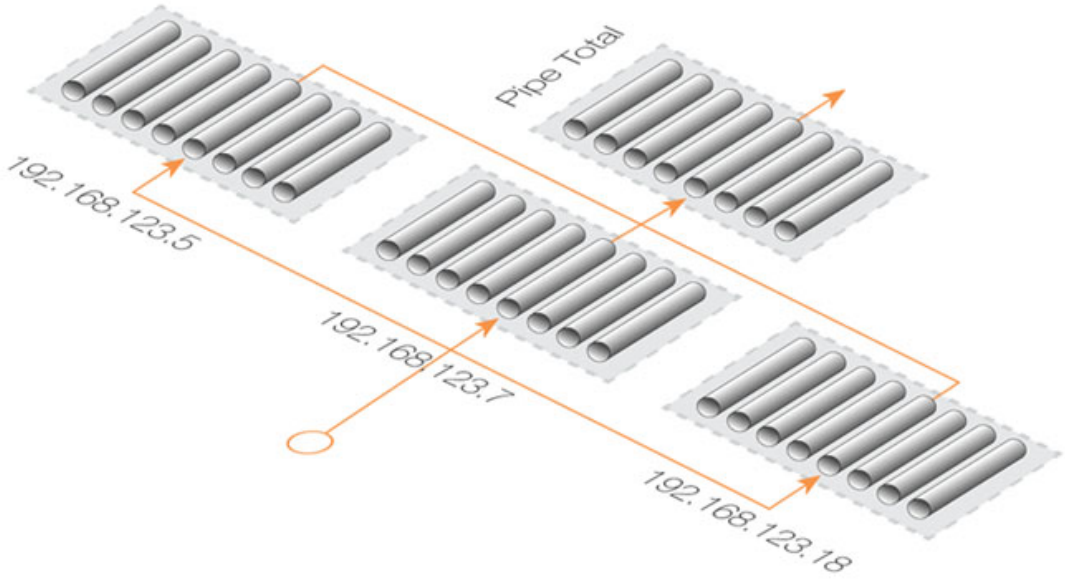
Note

*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** at the lowest precedence only and hence compete for the 250 kbps of available bandwidth with other traffic.*

11.1.9. Groups

CorePlus provides further granularity of control within pipes through the ability to split pipe bandwidth according to either the packet's source/destination network, IP, port or interface. This is referred to as creating *Groups* where the members of a group, sometimes called the *users*, can have limits and guarantees applied to them. The most common usage of this division of traffic is to group by IP or interface.

Figure 11.4. Traffic grouped per IP address.



If grouping by port is used then this implicitly also includes the IP address so that port 1024 of computer A is not the same as port 1024 of computer B and individual connections are identifiable. If grouping by network is chosen, the network size should be also be specified (this has the same meaning as the netmask).

A Simple Groups Scenario

If the total bandwidth limit for a pipe is 400 bps and we want to allocate this bandwidth amongst many destination IP addresses so no one IP address can take more then 100 bps of bandwidth, we select "Per DestIP" grouping and enter the total limit for the grouping as 100 bps. Bandwidth is then allocated on a "first come, first forwarded" basis but no one destination IP address can ever take more than 100 bps. No matter how many connections are involved the combined total bandwidth can still not exceed the pipe limit of 400 bps.

Instead of specifying a total group limit, the alternative is to enable the *Dynamic Balancing* option. This ensures that the available bandwidth is divided equally between all addresses regardless of how many there are and this is done up to the limit of the pipe. If a total group limit of 100 bps is also specified, as before, then no one user may take more than that amount of bandwidth.

Group Limits and Guarantees

In addition to specifying a total limit for group users, limits can be specified for each preference. If we specify a group user limit of 30 bps for precedence 2 then this means that users assigned a precedence of 2 by a Pipe Rule will be guaranteed 30 bps no matter how many users use the pipe. Just as with normal pipe precedences, traffic in excess of 30 bps for users at precedence 2 is moved down to the Best Effort precedence.

Continuing with the previous example, we could limit how much guaranteed bandwidth each inside user gets for inbound SSH traffic. This prevents a single user from using up all available high-priority bandwidth.

First we group the users of the **ssh-in** pipe so limits will apply to each user on the internal network. Since the packets are inbound, we select the grouping for the **ssh-in** pipe to be "Per DestIP".

Now we specify per-user limits by setting the precedence 2 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 desired, we could also limit the group total bandwidth for each user to some value, such as 40 kbps.

There will be a problem if there are more than 5 users utilizing SSH simultaneously: 16 kbps times 5 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 2 bandwidth the same way they have to compete for the lowest precedence bandwidth. Some users will still get their 16 kbps, some will not.

Dynamic balancing can be enabled to improve this situation by making sure all of the 5 users get the same amount of limited bandwidth. When the 5th user begins to generate SSH traffic, balancing lowers the limit per user to about 13 kbps (64 kbps divided by 5 users).

Dynamic 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.

11.1.10. Recommendations

The importance of setting a pipe limit

Traffic shaping only comes into effect when a CorePlus pipe is *full*. That is to say, it is passing as much traffic as the total limit allows. If a 500 kbps pipe is carrying 400 kbps of low priority traffic and 90 kbps of high priority traffic then there is 10 kbps of bandwidth left and there is no reason to throttle back anything. It is therefore important to specify a total limit for a pipe so that it knows what its capacity is and the precedence mechanism is totally dependent on this.

Pipe limits for VPN

Traffic shaping measures the traffic inside VPN tunnels. This is the raw unencrypted data without any protocol overhead so it will be less than the actual VPN traffic. VPN protocols such as IPsec can add significant overhead to the data and for this reason it is recommended that the limits

specified in the traffic shaping pipes for VPN traffic are set at around 20% below the actual available bandwidth.

Relying on the group limit

A special case when a total pipe limit is not specified is when a group limit is used instead. The bandwidth limit is then placed on, for example, each user of a network where the users must share a fixed bandwidth resource. An ISP might use this approach to limit individual user bandwidth by specifying a "Per DestinationIP" grouping. Knowing when the pipe is full is not important since the only constraint is on each user. If precedences were used the pipe maximum would have to be used.

Limits should not be higher than the available bandwidth

If pipe limits are set higher than the available bandwidth, the pipe will not know when the physical connection has reached its capacity. If the connection is 500 kbps but the total pipe limit is set to 600 kbps, the pipe will believe that it is not full and it will not throttle lower precedences.

Limits should be slightly less than available bandwidth

Pipe limits should be slightly below the network bandwidth. A recommended value is to make the pipe limit 95% of the physical limit. The need for this difference becomes less with increasing bandwidth since 5% represents an ever larger piece of the total.

The reason for the lower pipe limit is how CorePlus processes traffic. For outbound connections where packets leave the Clavister Security Gateway, there is always the possibility that CorePlus might slightly overload the connection because of the software delays involved in deciding to send packets and the packets actually being dispatched from buffers.

For inbound connections, there is less control over what is arriving and what has to be processed by the traffic shaping subsystem and it is therefore more important to set pipe limits slightly below the real connection limit to account for the time needed for CorePlus to adapt to changing conditions.

Attacks on Bandwidth

Traffic shaping cannot protect against incoming resource exhaustion attacks, such as DoS attacks or other flooding attacks. CorePlus will prevent these extraneous packets from reaching the hosts behind the Clavister Security Gateway, but cannot protect the connection becoming overloaded if an attack floods it.

Watching for Leaks

When setting out to protect and shape a network bottleneck, make sure that all traffic passing through that bottleneck passes through the defined CorePlus pipes.

If there is traffic going through your Internet connection that the pipes do not know about, they cannot 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 without being measured by pipes will have the same effect as bandwidth consumed by parties outside of administrator control but sharing the same connection.

Troubleshooting

For a better understanding of what is happening in a live setup, the console command:

```
pipe -u <pipename>
```

can be used to display a list of currently active users in each pipe.

11.1.11. A Summary of Traffic Shaping

CorePlus traffic shaping provides a sophisticated set of mechanisms for controlling and prioritising network packets. The following points summarize its use:

- Select the traffic to manage through *Pipe Rules*.
- Pipe Rules send traffic through *Pipes*.
- A pipe can have a limit which is the maximum amount of traffic allowed.
- A pipe can only know when it is *full* if a limit is specified.
- A single pipe should handle traffic in only one direction (although 2 way pipes are allowed).
- Pipes can be chained so that one pipe's traffic feeds into another pipe.
- Specific traffic types can be given a *priority* in a pipe
- Priorities can be given a maximum limit which is also a guarantee. Traffic that exceeds this will be sent at the minimum precedence which is also called the *Best Effort* precedence.
- At the Best Effort precedence all packets are treated on a "first come, first forwarded" basis.
- Within a pipe, traffic can also be separated on a *Group* basis. For example, by source IP address. Each user in a group (for example, each source IP address) can be given a maximum limit and precedences within a group can be given a limit/guarantee.
- A pipe limit need not be specified if group members have a maximum limit.
- *Dynamic Balancing* can be used to specify that all users in a group get a fair and equal amount of bandwidth.

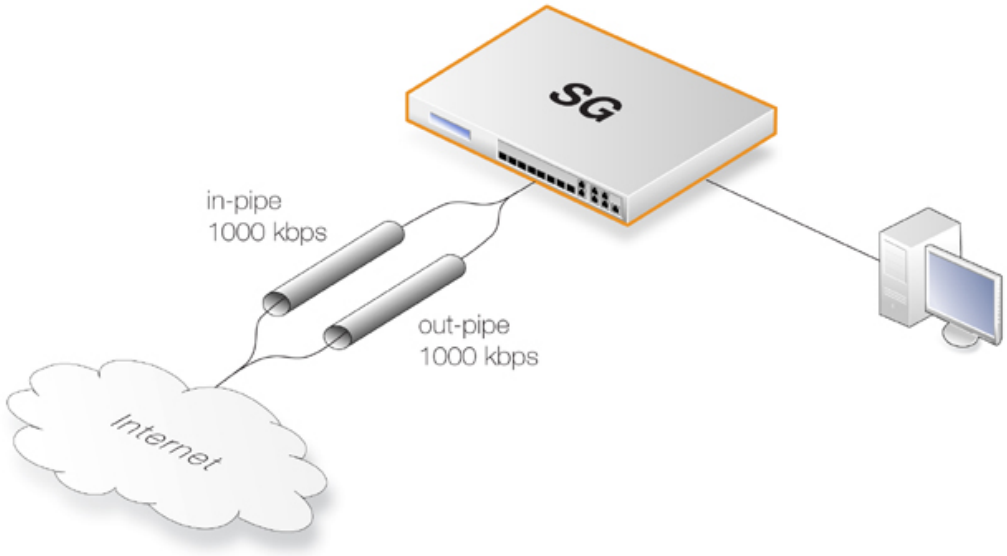
11.1.12. More Pipe Examples

This section looks at some more scenarios and how traffic shaping can be used to solve particular problems.

A Basic Scenario

The first scenario will examine the configuration shown in the image below, in which incoming and outgoing traffic is to be limited to 1 megabit per second.

Figure 11.5. A Basic Traffic Shaping Scenario



The reason for using 2 different pipes in this case, is that these are easier to match to the physical link capacity. This is especially true with asynchronous links such as ADSL.

First, two pipes called *in-pipe* and *out-pipe* need to be created with the following parameters:

Pipe Name	Min Prec	Def Prec	Max Prec	Grouping	Net size	Pipe limit
in-pipe	0	0	7	PerDestIP	24	1000kb
out-pipe	0	0	7	PerSrcIP	24	1000kb

Dynamic Balancing should be enabled for both pipes. Instead of *PerDestIP* and *PerSrcIP* we could have used *PerDestNet* and *PerSrcNet* if there were several networks on the inside.

The next step is to create the following Pipe Rule which will force traffic to flow through the pipes.

Rule Name	Forward Pipes	Return Pipes	Source Interface	Source Network	Destination Interface	Destination Network	Service
all_1mbps	out-pipe	in-pipe	lan	lannet	wan	all-nets	all

The rule will force all traffic to the default precedence level and the pipes will limit total traffic to their 1 Mbps limit. Having **Dynamic Balancing** enabled on the pipes means that all users will be allocated a fair share of this capacity.

Using Several Precedences

We now extend the above example by allocating priorities to different kinds of traffic accessing the internet from a headquarters office.

Lets assume we have a symmetric 2/2 Mbps link to the internet We will allocate descending priorities and traffic requirements to the following users:

- **Priority 6** - VoIP (500 kpbs)
- **Priority 4** - Citrix (250 kpbs)
- **Priority 2** - Other traffic (1000 kpbs)

- **Priority 0** - Web plus remaining from other levels

To implement this scheme, we can use the *in-pipe* and *out-pipe*. We first enter the *Pipe Limits* for each pipe. These limits correspond to the list above and are:

- **Priority 6** - *500*
- **Priority 4** - *250*
- **Priority 2** - *1000*

Now create the *Pipe Rules*:

Rule Name	Forward Pipes	Return Pipes	Source Interface	Source Network	Destination Interface	Destination Network	Service	Prec
web_surf	out-pipe	in-pipe	lan	lannet	wan	all-nets	http_all	0
voip	out-pipe	in-pipe	lan	lannet	wan	all-nets	H323	6
citrix	out-pipe	in-pipe	lan	lannet	wan	all-nets	citrix	4
other	out-pipe	in-pipe	lan	lannet	wan	all-nets	All	2

These rules are processed from top to bottom and force different kinds of traffic into precedences based on the *Service*. Customized service objects may need to be first created in order to identify particular types of traffic. The *all* service at the end, catches anything that falls through from earlier rules since it is important that no traffic bypasses the pipe rule set otherwise using pipes will not work.

Pipe Chaining

Suppose the requirement now is to limit the precedence 2 capacity (other traffic) to 1000 kbps so that it does not spill over into precedence 0. This is done with *pipe chaining* where we create new pipes called *in-other* and *out-other* both with a *Pipe Limit* of 1000. The *other* pipe rule is then modified to use these:

Rule Name	Forward Pipes	Return Pipes	Source Interface	Source Network	Destination Interface	Destination Network	Service	Prec
other	out-other out-pipe	in-other in-pipe	lan	lannet	wan	all-nets	All	2

Note that *in-other* and *out-other* are first in the pipe chain in both directions. This is because we want to limit the traffic immediately, before it enters the *in-pipe* and *out-pipe* and competes with Voip, Citrix and Web-surfing traffic.

A VPN Scenario

In the cases discussed so far, all traffic shaping is occurring inside a single Clavister Security Gateway. VPN is typically used for communication between a headquarters and branch offices in which case pipes can control traffic flow in both directions. With VPN it is the tunnel which is the source and destination interface for the pipe rules.

An important consideration which has been discussed previously, is allowance in the *Pipe Total* values for the overhead used by VPN protocols. As a rule of thumb, a pipe total of 1700 is reasonable for a VPN tunnel where the underlying physical connection capacity is 2 Mbps.

It is also important to remember to insert into the pipe all non-VPN traffic using the same physical link.

The *pipe chaining* can be used as a solution to the problem of VPN overhead. A limit which allows for this overhead is placed on the VPN tunnel traffic and non-VPN traffic is inserted into a pipe that matches the speed of the physical link.

To do this we first create separate pipes for the outgoing traffic and the incoming traffic. VoIP traffic will be sent over a VPN tunnel that will have a high priority. All other traffic will be sent at the *best effort* priority (see above for an explanation of this term). Again, we'll assume a 2/2 Mbps symmetric link.

The pipes required will be:

- **vpn-in**
  - *Priority 6:* VoIP 500 kpbs
  - *Priority 0:* Best effort

  Total: 1700
- **vpn-out**
  - *Priority 6:* VoIP 500 kpbs
  - *Priority 0:* Best effort

  Total: 1700
- **in-pipe**
  - *Priority 6:* VoIP 500 kpbs

  Total: 2000
- **out-pipe**
  - *Priority 6:* VoIP 500 kpbs

  Total: 2000

The following pipe rules are then needed to force traffic into the correct pipes and precedence levels:

Rule Name	Forward Pipes	Return Pipes	Src Int	Source Network	Dest Int	Destination Network	Service	Prec
vpn_voip_out	vpn-out out-pipe	vpn-in in-pipe	lan	lannet	vpn	vpn_remote_net	H323	6
vpn_out	vpn-out out-pipe	vpn-in in-pipe	lan	lannet	vpn	vpn_remote_net	All	0
vpn_voip_in	vpn-in in-pipe	vpn-out out-pipe	vpn	vpn_remote_net	lan	lannet	H323	6
vpn_in	vpn-in in-pipe	vpn-out out-pipe	vpn	vpn_remote_net	lan	lannet	All	0
out	out-pipe	in-pipe	lan	lannet	wan	all-nets	All	0
in	in-pipe	out-pipe	wan	all-nets	lan	lannet	All	0

With this setup, all VPN traffic is limited to 1700 kbps, the total traffic is limited to 2000 kbps and VoIP to the remote site is guaranteed 500 kbps of capacity before it is forced to best effort.

SAT with Pipes

If SAT is being used, for example with a web server or ftp server, that traffic also needs to be forced into pipes or it will escape traffic shaping and ruin the planned quality of service. In addition, server traffic is initiated from the outside so the order of pipes needs to be reversed: the forward pipe is the *in-pipe* and the return pipe is the *out-pipe*.

A simple solution is to put a "catch-all-inbound" rule at the bottom of the pipe rule. However, the external interface (*wan*) should be the source interface to avoid putting into pipes traffic that is coming from the inside and going to the external IP address. This last rule will therefore be:

Rule Name	Forward Pipes	Return Pipes	Source Interface	Source Network	Destination Interface	Destination Network	Service	Prec
all-in	in-pipe	out-pipe	wan	all-nets	core	all-nets	All	0

Note

*If the SAT is from an ARPed IP address, the **wan** interface needs to be the destination.*

11.2. Threshold Rules

11.2.1. Overview

The objective of a *Threshold Rule* 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. It might alternatively be some external source trying to open excessive numbers of connections. (A "connection" in this context refers to all types of connections, such as TCP, UDP or ICMP, tracked by the CorePlus state-engine).

A Threshold Rule is like a normal policy based rule. A combination of source/destination network/interface can be specified for a rule and a type of service such as HTTP can be associated with it. Each rule can have associated with it one or more **Action**s which specify how to handle different threshold conditions.

A Threshold has the following parameters:

- **Action** - The response to exceeding the limit: either **Audit** or **Protect**
- **Group By** - Either **Host** or **Network** based
- **Threshold** - The numerical limit which must be exceeded to trigger a response
- **Threshold Type** - Limiting connections per second or limiting total number of concurrent connections

These parameters are described below:

11.2.2. Connection Rate/Total Connection Limiting

Connection Rate Limiting allows an administrator to put a limit on the number of new connections being opened to the Clavister Security Gateway per second.

Total Connection Limiting allows the administrator to put a limit on the total number of connections opened to the Clavister Security Gateway. This function is extremely useful when NAT pools are required due to the large number of connections generated by P2P users.

11.2.3. Grouping

The two groupings are as follows:

- **Host Based** - The threshold is applied separately to connections from different IP addresses.
- **Network Based** - The threshold is applied to all connections matching the rules as a group.

11.2.4. Rule Actions

When a Threshold Rule is triggered one of two responses are possible:

- **Audit** - Leave the connection intact but log the event
- **Protect** - Drop the triggering connection

Logging would be the preferred option if the appropriate triggering value cannot be determined beforehand. Multiple Actions for a given rule might consist of **Audit** for a given threshold while the action might become **Protect** for a higher threshold.

11.2.5. Multiple Triggered Actions

When a rule is triggered then CorePlus will perform the associated rule Actions that match the condition that has occurred. If more than one Action matches the condition then those matching Actions are applied in the order they appear in the user interface.

If several Actions that have the same combination of **Type** and **Grouping** (see above for the definition of these terms) are triggered at the same time, only the Action with the highest threshold value will be logged

11.2.6. Exempted Connections

It should be noted that some advanced settings known as *Before Rules* settings can exempt certain types of connections for remote management from examination by the CorePlus rule set. These settings will also exempt the connections from Threshold Rules.

11.2.7. Threshold Rule Blacklisting

If the **Protect** option is used, Threshold Rules can be configured so that the source that triggered the rule, is added automatically to a *Blacklist* of IP addresses or networks. If several **Protect** Actions with blacklisting enabled are triggered at the same time, only the first triggered blacklisting Action will be executed by CorePlus.

A host based Action with blacklisting enabled will blacklist a single host when triggered. A network based action with blacklisting enabled will blacklist the source network associated with the rule. If the Threshold Rule is linked to a service then it is possible to block only that service.

When Blacklisting is chosen, then the administrator can elect that existing connections from the triggering source can be left unaffected or they can be dropped.

The length of time, in seconds, for which the source is blacklisted can also be set.

This option is discussed further in Section 7.7, “Blacklisting Hosts and Networks”.

11.3. Server Load Balancing

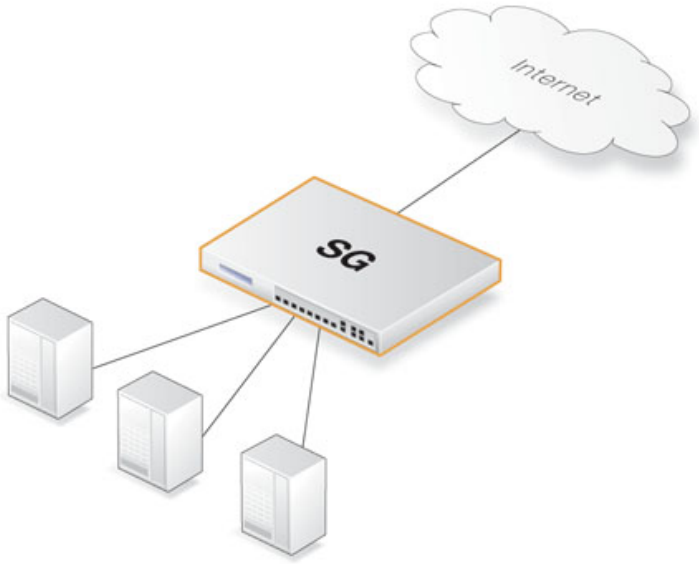
11.3.1. Overview

The *Server Load Balancing* (SLB) feature in CorePlus is a powerful tool that can improve the following aspects of network applications:

- Performance
- Scalability
- Reliability
- Ease of administration

SLB allows network service demands to be shared among multiple servers. This improves both the performance and the scalability applications by allowing a cluster of multiple servers (sometimes called a "server farm") to handle many more requests than a single server. The image below illustrates a typical SLB scenario, with Internet access to applications being controlled by the Clavister Security Gateway.

Figure 11.6. A Server Load Balancing Configuration



Besides 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 or has less load.

SLB also means that network administrators can perform maintenance tasks 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 a server farm, or taking down applications.

The combination of network monitoring and distributed load sharing also provides an extra level of protection against Denial Of Service (DoS) attacks.

CorePlus SLB is implemented through the use of *SLB_SAT* rules in the IP rule set and these rules offer administrators a choice of several different algorithms to distribute the load. This allows the

tailoring of SLB to best suit the needs of the network.

There are four issues to be considered when using SLB:

1. The target servers across which the load is to be balanced
2. The load distribution mode
3. The SLB algorithm used
4. The monitoring method

Each of these topics is discussed further in the sections that follow.

11.3.2. Identifying the Servers

The first step is to identify the servers across which the load is to be balanced. This might be a *server farm* which is a cluster of servers set up to work as a single "virtual server". The servers that are to be treated as a single vitual server by SLB must be specified.

11.3.3. The Load Distribution Mode

No single method of distributing the server load is ideal for all services. Different types of services have different needs. In the IP rule set the administrator can configure rules for specific services. SLB will then filter the packet flow according to these rules.

CorePlus SLB supports the following distribution modes:

Per-state Distribution — In this mode, 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.

IP Address Stickiness — 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.

Network Stickiness — This mode is similar to IP stickiness except that by using a subnet mask, a range of hosts in a subnet can be specified.

11.3.4. The Distribution Algorithm

There are several ways to determine how a load is shared across a server farm. CorePlus SLB supports the following algorithms:

Round Robin — The algorithm distributes new incoming connections to a list of servers on a rotating basis. For the first connection, the algorithm picks a server 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 take turns in being assigned the next 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. 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.

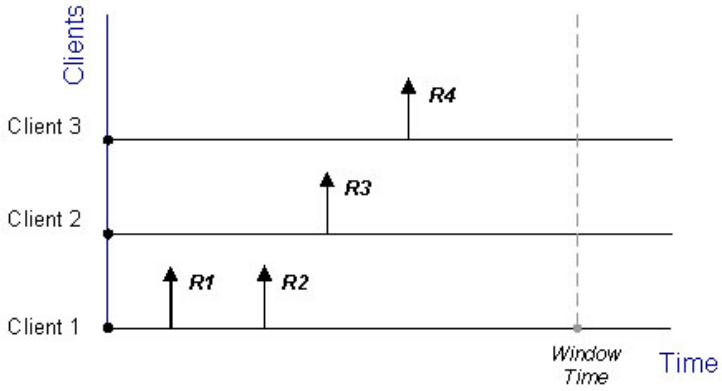
If the Connection Rate algorithm is used without stickiness, it will behave as a Round Robin algorithm that allocates new connections to servers in an orderly fashion. It will also behave as the Round Robin algorithm if there are always clients with a new IP address that make one connection.

The real benefit is when using Connection Rate together with stickiness and clients make multiple connections. Connection Rate will then ensure that the distribution of new connections is as even as possible among 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 do not get as many new connections.

In the management interface, the time window is variable for counting the number of seconds back in time to summarize the number of new connections for the 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 remembered.

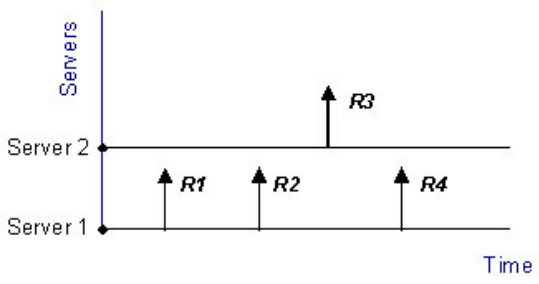
An example is shown in the figure below. In this example, the Clavister Security Gateway is responsible for balancing connections from 3 clients with different addresses to 2 servers. Stickiness is set.

Figure 11.7. Connections from Three Clients



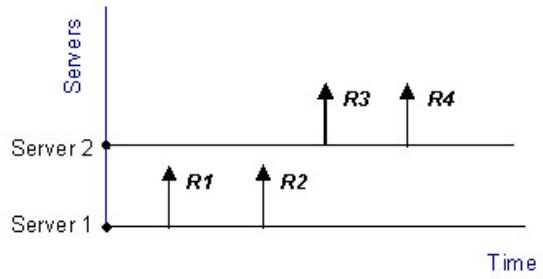
When the Round Robin algorithm is used, the first arriving requests R1 and R2 from Client 1 are both assigned to one sever, say Server 1, according to stickiness. The next request R3 from Client 2 is then routed to Server 2. When R4 from Client 3 arrives, Server 1 gets back its turn again and will be assigned with R4.

Figure 11.8. 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 11.9. Stickiness and Connection Rate



Regardless which algorithm is chosen, if a server goes down, traffic will be sent to other servers. And when the server comes back online, it can automatically be placed back into the server farm and start getting requests again.

11.3.5. Server Health Monitoring

SLB uses *Server Health Monitoring* to continuously check the condition of the servers in an SLB configuration. SLB can monitor different OSI layers to check the condition of each server. Regardless of the algorithms used, if a server is deemed to have failed, SLB will not open any more connections to it until the server is restored to full functionality.

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. 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. 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 SLB does not receive a TCP SYN/ACK back, it will mark port 80 on that server as down. SLB recognizes the conditions *no response*, *normal response* or *closed port response* from servers.

11.3.6. SLB_SAT Rules

The key component in setting up SLB is the *SLB_SAT* rule in the IP rule set. The steps that should be followed are:

1. Define an Object for each server for which SLB is to be done.
2. Define a Group which included all these objects
3. Define an SLB_SAT Rule in the IP rule set which refers to this Group and where all other SLB parameters are defined.
4. Define a further rule that duplicates the source/destination interface/network of the SLB_SAT rule that allows traffic through. The could be one or combination of
   - ForwardFast

- Allow
- NAT

The table below shows the rules that would be defined for a typical scenario of a set of webservers behind the Clavister Security Gateway for which the load is being balanced. The **Allow** rule allows external clients to access the webservers.

Rule Name	Rule Type	Src. Interface	Src. Network	Dest. Interface	Dest. Network
WEB_SLB	SLB_SAT	any	all-nets	core	ip_ext
WEB_SLB_ALW	Allow	any	all-nets	core	ip_ext

If there are clients on the same network as the webservers that also need access to those webservers then an **NAT** rule would also be used:

Rule Name	Rule Type	Src. Interface	Src. Network	Dest. Interface	Dest. Network
WEB_SLB	SLB_SAT	any	all-nets	core	ip_ext
WEB_SLB_NAT	NAT	lan	lannet	core	ip_ext
WEB_SLB_ALW	Allow	any	all-nets	core	ip_ext

Note that the destination interface is specified as **core**, meaning CorePlus itself deals with this. The key advantage of having a separate **Allow** rule is that the webservers can log the exact IP address that is generating external requests. Using only a **NAT** rule, which is possible, means that webservers would see only the IP address of the Clavister Security Gateway

Example 11.3. Setting up SLB

In this example server load balancing is to be done between 2 HTTP webservers which are situated behind the Clavister Security Gateway. The 2 webservers have the private IP addresses *192.168.1.10* and *192.168.1.11* respectively. The default SLB values for monitoring, distribution method and stickiness are used.

A NAT rule is used in conjunction with the SLB_SAT rule so that clients behind the gateway can access the webservers. An **Allow** rule is used to allow access by external clients.

Web Interface

A. Create an Object for each the webservers:

1. Go to **Objects > Address Book > Add > IP Address**
2. Enter a suitable name, for example *server1*
3. Enter the **IP Address** as *192.168.1.10*
4. Click **OK**
5. Repeat the above to create an object called *server2* for the *192.168.1.11* IP address.

B. Create a Group which contains the 2 webserver objects:

1. Go to **Objects > Address Book > Add > IP4 Group**
2. Enter a suitable name, for example *server_group*
3. Add *server1* and *server2* to the group
4. Click **OK**

C. Specify the **SLB_SAT** IP rule:

1. Go to **Rules > IP Rule Sets > main > Add > IP Rule**

2. Enter:
   - **Name:** Web_SLB
   - **Action:** SLB_SAT
   - **Service:** HTTP
   - **Source Interface:** any
   - **Source Network:** all-nets
   - **Destination Interface:** core
   - **Destination Network:** ip_ext
3. Select tab **SAT SLB**
4. Under **Server Addresses** add *server_group* to **Selected**
5. Click **OK**

D. Specify a matching **NAT** IP rule for internal clients:

1. Go to **Rules > IP Rule Sets > main > Add > IP Rule**
2. Enter:
   - **Name:** Web_SLB_NAT
   - **Action:** NAT
   - **Service:** HTTP
   - **Source Interface:** lan
   - **Source Network:** lannet
   - **Destination Interface:** core
   - **Destination Network:** ip_ext
3. Click **OK**

E. Specify an **Allow** IP rule for the external clients:

1. Go to **Rules > IP Rule Sets > main > Add > IP Rule**
2. Enter:
   - **Name:** Web_SLB_ALW
   - **Action:** Allow
   - **Service:** HTTP
   - **Source Interface:** any
   - **Source Network:** all-nets
   - **Destination Interface:** core
   - **Destination Network:** ip_ext
3. Click **OK**

Chapter 12. High Availability

This chapter describes the high availability fault-tolerance feature in Clavister Security Gateways.

12.1. Overview

HA Clusters

Clavister *High Availability* (HA) works by adding a back-up *slave* Clavister Security Gateway to an existing *master* security gateway. The master and slave are connected together and make up a logical *HA Cluster*. One of the units in a cluster will be *active* when the other unit is *inactive* and on standby. Initially the slave will be inactive and will monitor the master. If the slave detects that the master is not responding, a *failover* takes place and the slave becomes active. If the master later regains full functionality the slave will continue to be active, with the master now monitoring the slave and failover only taking place if the slave fails. This is sometimes known as an *active-passive* HA implementation.

The *Master* and *Active* Units

It should be kept in mind that the *master* unit in a cluster is not always the same as the *active* unit. The *active* unit is the Clavister Security Gateway that is processing all traffic at a given point in time. This could be the *slave* if a failover has occurred because the *master*'s operation has been impaired.

Inter-connection

In a cluster, the master and slave units must be directly connected to each other by a synchronization connection which is known to CorePlus as the **sync** interface. One of the normal interfaces on the master and the slave are dedicated for this purpose and are connected together with a crossover cable.

Packets, known as *heartbeats*, are continually sent across the **sync** and all other interfaces from one unit to the other so that peer health can be monitored and this mechanism is discussed further in the section that follows. These packets are sent in both directions so that the passive unit knows about the health of the active unit and and the active knows about the passive.

Cluster Management

An HA Cluster of two Clavister Security Gateways is managed as a single unit with a unique cluster name which appears in the management interface as a single logical Clavister Security Gateway. Administration operations such as changing rules in the IP rule set are carried out as normal with the changes automatically being made to the configurations of both the master and the slave.

Load-sharing

Clavister HA clusters do not provide load-sharing since only one unit will be active while the other is inactive and only two Clavister Security Gateways, the master and the slave, can exist in a single cluster. The only processing role that the inactive unit plays is to replicate the state of the active unit and to take over all traffic processing if it detects the active unit is not responding.

Hardware Duplication

Clavister HA will only operate between two Clavister Security Gateways. As the internal operation of different security gateway manufacturer's software is completely dissimilar, there is no common method available to communicating state information to a dissimilar device.

It is also strongly recommended that the Clavister Security Gateways used in cluster have identical configurations. They must also have identical licenses which allow identical capabilities including the ability to run in an HA cluster.

Extending Redundancy

Implementing an HA Cluster will eliminate one of the points of failure in a network. Routers, switches and Internet connections can remain as potential points of failure and redundancy for these should also be considered.

Protecting Against Network Failures Using HA and Link Monitor

The CorePlus *Link Monitor* feature can be used to check connection with a host so that when it is no longer reachable an HA failover is initiated to a peer which has a different connection to the host. This technique is a useful extension to normal HA usage which provides protection against network failures between a single Clavister Security Gateway and hosts. This technique is described further in Section 3.4.3, “The Link Monitor”.

The following sections describe the High Availability feature in greater detail.

12.2. HA Mechanisms

Basic Principles

Clavister HA provides a redundant, state-synchronized hardware configuration. The state of the active unit, such as the connection table and other vital information, is continuously copied to the inactive unit via the **sync** interface. When cluster failover occurs, the inactive unit knows which connections are active, and traffic can continue to flow.

The inactive system detects that the active system is no longer operational when it no longer detects sufficient *Cluster Heartbeats*. Heartbeats are sent over the **sync** interface as well as all other interfaces. CorePlus sends 5 heartbeats per second from the active system and when three heartbeats are missed (that is to say, after 0.6 seconds) a failover will be initiated. By sending heartbeats over all interfaces, the inactive unit gets an overall view of the active unit's health. Even if **sync** is deliberately disconnected, failover may not result if the inactive unit receives enough heartbeats from other interfaces via a shared switch, however the **sync** interface sends twice as many heartbeats as any of the normal interfaces. The administrator can disable heartbeat sending on any of the interfaces.

Heartbeats are not sent at smaller intervals because such delays may occur during normal operation. An operation such as opening a file, could result in delays long enough to cause the inactive system to go active, even though the other is still active.

Heartbeat Characteristics

Cluster heartbeats have the following characteristics:

- The source IP is the interface address of the sending security 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 therefore cannot be trusted.
- It is a UDP packet, sent from port 999, to port 999.
- The destination MAC address is the ethernet multicast address corresponding to the shared hardware address. In other words, *11-00-00-C1-4A-nn*. Link-level multicasts are used over normal unicast packets for security: using unicast packets would mean that a local attacker could fool switches to route heartbeats somewhere else so the inactive system never receives them.

Failover Time

The time for failover is typically about one second which means that clients may experience a failover as a slight burst of packet loss. In the case of TCP, the failover time is well within the range of normal retransmit timeouts so TCP will retransmit the lost packets within a very short space of time, and continue communication. UDP does not allow retransmission since it is inherently an unreliable protocol.

Shared IP Addresses and ARP

Both master and slave 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, are answered by the active system. The hardware address of the shared IP address and other published addresses are not related to the actual hardware addresses of the interfaces. Instead the MAC address is constructed by CorePlus from the Cluster ID in the following form: 10-00-00-C1-4A-nn, where nn comes from combining the Cluster ID configured in the Advanced Settings section with the

hardware bus/slot/port of the interface. The Cluster ID must be unique for each cluster in a network.

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 cluster member discovers that its peer is not operational, it broadcasts gratuitous ARP queries on all interfaces using the shared hardware address as the sender address. This allows switches to re-learn within milliseconds where to send packets destined for the shared address. The only delay in failover therefore, is detecting that the active unit is down.

ARP queries are also broadcast periodically to ensure that switches do not forget where to send packets destined for the shared hardware address.

HA with Anti-Virus and IDP

If a CorePlus cluster has the Anti-Virus or IDP subsystems enabled then updates to the Anti-Virus signature database or IDP pattern database will routinely occur. These updates involve downloads from the external Clavister databases and they require CorePlus reconfiguration to occur for the new database contents to become active. Database updates cause the following sequence of events to occur in an HA cluster:

1. The active (master) unit downloads the new database files from the Clavister servers.
2. The active (master) node sends the new database file to the inactive peer.
3. The inactive (slave) unit reconfigures to activate the new database files.
4. The active (master) unit now reconfigures to activate the new database files causing a failover to the slave unit. The slave is now the active unit.
5. After reconfiguration of the master is complete, failover occurs again so that the master once again becomes the active unit.

12.3. HA Setup

This section provides a step-by-step guide for setting up an HA Cluster. The setup process can be simplified using the *HA Setup Wizard* as described below.

12.3.1. Hardware Setup

1. Start with two physically similar Clavister Security Gateways. Both may be newly purchased or one may have been purchased to be the back-up unit (in other words, to be the slave unit).

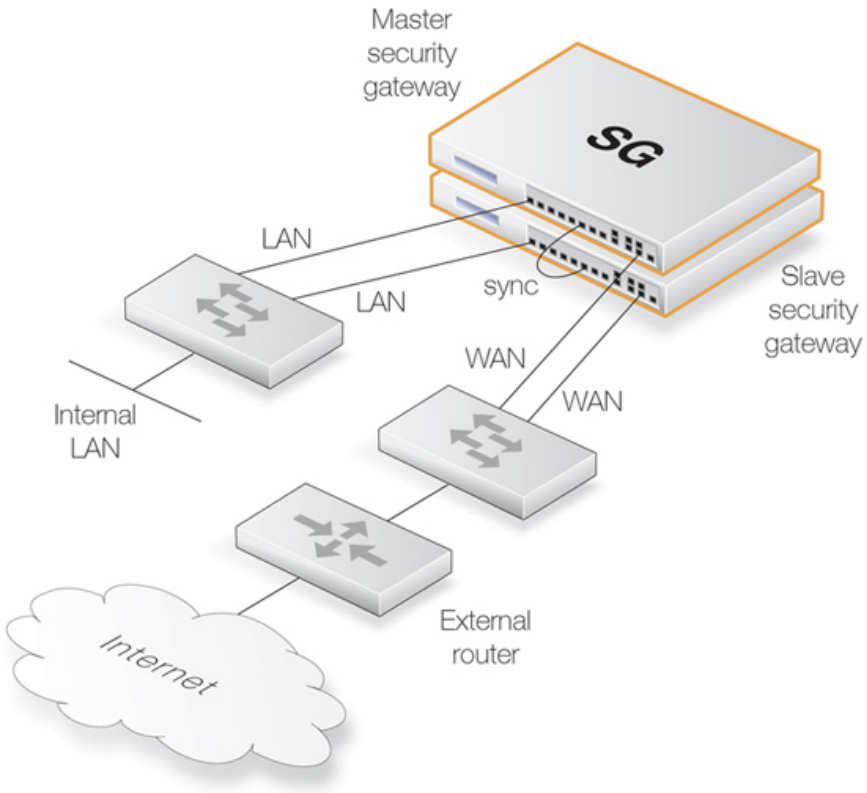
2. There should be two valid CorePlus cluster licenses, one for the master and one for the slave. These should have identical capabilities and both should have the cluster option enabled.

3. Make the physical connections:

   - Connect the matching interfaces of master and slave through a common switch.

   - Select an interface on the master and slave which is to be used by the units for monitoring each other and connect them together with an Ethernet crossover cable. This will be the CorePlus **sync** interface. It is recommended that the same interface is used on both master and slave, assuming they are similar systems.

Figure 12.1. High Availability Setup



The illustration above shows the typical HA Cluster connections. All interfaces of the master would normally also be present on the slave and be connected to the same networks. This is achieved by connecting the same interfaces on both master and slave via a switch to other network portions.

The **lan** interface on the master and the **lan** interface on the slave would be connected to the same switch which then connects to an internal network. Similarly the **wan** interface on the master and the **wan** interface would connect to a switch which in turn connects to the external Internet.

The hardware of the slave does not need to exactly match the master, however it is recommended that hardware with similar performance is used in order to avoid any throughput degradation after a failover.

4. Decide on a shared IP address for each interface in the cluster. Some interfaces could have shared addresses only while others could also have unique, individual IP addresses for each interface specified in a *IP4 HA Address* object. The shared and individual addresses are used as follows:

   - The individual addresses specified for an interface in an IP4 HA Address object allow remote management through that interface. These addresses can also be "pinged" using ICMP. If either unit is inpoerative, its individual IP addresses will also be unreachable. These IP addresses are usually private but must be public if management access across the public Internet is required.

     If an interface is not assigned an individual address through an IP4 HA Address object then it must be assigned the default address *localhost* which is an IP address from the subnet *127.0.0.0/8*.

     ARP queries for the individual IP addresses specified in IP4 HA Address objects are answered by the security gateway that owns the address, using the normal hardware address, just as with normal IP units.

   - One single shared IP address is used for routing and it is also the address used by dynamic address translation, unless the configuration explicitly specifies another address.

Note

The shared IP address cannot be used for remote management or monitoring purposes. When using, for example, SSH for remote management of the Clavister Security Gateways in an HA Cluster, the individual IP addresses of the security gateway's interfaces must be used and these are specified in IP4 HA Address objects as discussed above.

Wizard and Manual Software Setup

The software setup procedures are now divided into the two sections that follow:

- Section 12.3.2, “CorePlus Setup with the HA Wizard” for fast, simple setup.
- Section 12.3.3, “CorePlus Manual HA Setup” for step by step manual setup, without the wizard.

12.3.2. CorePlus Setup with the HA Wizard

CorePlus provides a wizard to automate the HA setup procedure. The wizard needs to be run twice: once when connected to the master unit in the HA cluster, and a second time when connected to the slave unit in the cluster. The procedure for doing this with each unit is as follows:

1. Connect to the Clavister Security Gateway through the WebUI.

2. Go to **Objects** > **Address Book** and create an **IP4 HA Address** object for each interface pair in the cluster. Each object will contain the master and slave IP addresses for the interface pair. Inclusion in an IP4 HA Address object is mandatory for any interface that will be used for remote management but optional for other interfaces.

3. **Save and activate** the new configuration before continuing to the next step.

4. Go to the **HA** section of the user interface and press the **Start Wizard** button. Running the wizard for the master and slave units is described in *A* and *B* below.

5. **Save and activate** the new configuration.

A. Running the Wizard for Master Setup

1. Specify the **NodeType**, **ClusterID** and interface that will be used for synchronization.

2. If the two hardware models used in the cluster are different, this should also be indicated (see Section 12.3.5, "Using Unique Shared Mac Addresses" below for an explanation of this option).

3. Select the shared IP address and, if desired, the individual IP4 HA Address objects created earlier.

4. The wizard will confirm that master setup is complete.

B. Running the Wizard for Slave Setup

1. Specify the **NodeType**, **ClusterID** and interface that will be used for synchronization.

2. If the two hardware models used in the cluster are different, this should also be indicated (see Section 12.3.5, "Using Unique Shared Mac Addresses" below for an explanation of this option).

3. The wizard will acknowledge that it will synchronize with the master unit.

4. Select the desired interface mapping.

5. The wizard will confirm that slave setup and synchonization is complete.

12.3.3. CorePlus Manual HA Setup

To set up an HA cluster manually, without the wizard, the steps are as follows:

1. Connect to the master unit with the WebUI.

2. Go to **System > High Availability**.

3. Check the **Enable High Availability** checkbox.

4. Set the **Cluster ID**. This must be unique for each cluster.

5. Choose the **Sync Interface**.

6. Select the node type to be *Master*.

7. Go to **Objects > Address Book** and create an **IP4 HA Address** object for each interface pair. Each must contain the master and slave interface IP addresses for the pair.

   Creating an object is mandatory for an interface pair used for remote management, but optional for other interfaces (in which case the default address *localhost* must be used which is an IP from the *127.0.0.0/8* subnet).

8. Go to **Interfaces > Ethernet** and go through each interface in the list, entering the shared IP address for that interface in the **IP Address** field.

   Also select the **Advanced** tab for each interface and set the **High Availability, Private IP**

Address field to be the name of the IP4 HA Address object created previously for the interface (CorePlus will automatically select the appropriate address from the master and slave addresses defined in the object).

Note

*The term **private IP address** is not strictly correct when used here. Either address used in an IP4 HA Address object may be public if management access across the public Internet is required.*

9. **Save and activate** the new configuration.

10. Repeat the above steps for the other Clavister Security Gateway but this time select the node type to be *Slave*.

Making Cluster Configuration Changes

The configuration on both Clavister Security Gateways needs to be the same. The configurations of the two units will be automatically synchronized. To change something in a cluster configuration, log on to either the master or the slave, make the change, then save and activate. The change is automatically made to both units.

12.3.4. Verifying that the Cluster Functions

To verify that the cluster is performing correctly, first use the *ha* command on each unit. The output will look similar to the following for the master:

```
Device:/> ha

This device is an HA MASTER
This device is currently ACTIVE (will forward traffic)
HA cluster peer is ALIVE
```

Then use the *stat* command to verify that both the master and slave have about the same number of connections. The output from the command should contain a line similar to the following:

```
Connections 2726 out of 128000
```

The lower number on the left in this output is the current number of connections and the higher number on the right is the maximum number of connections allowed by the license.

The following points are also relevant to cluster setup:

- If this is not the first cluster in a network then the *Cluster ID* must be changed for the cluster so that it is unique (the default value is *0*). The *Cluster ID* determines that the MAC address for the cluster is unique.

- Enabling the advanced setting *Use Unique Share MAC* is recommended so that each interface has its own MAC address. If this is not enabled, interfaces share a MAC address and this can confuse some third party switches.

- Make sure that the advanced setting *High Buffers* is set to be *automatic* for both units in the cluster. This setting determines how memory is allocated by CorePlus for handling increasing numbers of connections. A hardware restart is required for a change in this setting to take effect.

  Where a cluster has a very high number (for example, tens of thousands) of simultaneous connections then it may be necessary to set a high value for this instead of using *automatic*. A very high value for *High Buffers* can suit situations with large numbers of connections but can have the disadvantage of increasing throughput latency.

12.3.5. Using Unique Shared Mac Addresses

For HA setup, CorePlus provides the advanced option *Use Unique Shared MAC Address*. By default, this is enabled and in most configurations it should not need to be disabled. The effect of enabling this setting is that a single, unique MAC address will be used for each pair of matching hardware interfaces so that, for example, the *lan1* interface on the master unit will appear to have the same MAC address as the *lan1* interface on the slave unit.

An HA cluster will function if this setting is disabled but can cause problems with a limited number of switch types where the switch uses a shared ARP table. Such problems can be hard to diagnose which is why it is best to always have the setting enabled.

In one situation, this setting should be disabled and that is when an HA cluster is set up using non-matching hardware. In order to function correctly, unique shared MAC addresses should not be used. For this reason, the setup wizard asks if non-matching hardware is being used and adjusts this setting appropriately.

12.4. HA Issues

The following points should be kept in mind when managing and configuring an HA Cluster.

Real-time Monitoring

The Real-time Monitor will not automatically track the active Clavister Security Gateway. If a Real-time Monitor graph shows nothing but the connection count moving, then the cluster has probably failed over to the other unit.

SNMP

SNMP statistics are not shared between master and slave. SNMP managers have no failover capabilities. Therefore both security gateways in a cluster need to be polled separately.

Logging

Log data will be coming from both master and slave. This means that the log receiver will have to be configured to receive logs from both. It also means that all log queries will likely have to include both master and slave as sources which will give all the log data in one result view. Normally, the inactive unit will not be sending log entries about live traffic so the output should look similar to that from one Clavister Security Gateway.

Using Individual IP Addresses

The unique individual IP addresses of the master and slave cannot safely be used for anything but management. 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 security gateway they belong to does.

Failed Interfaces

Failed interfaces will not be detected unless they fail to the point where CorePlus cannot continue to function. This means that failover will not occur if the active unit can still send "am alive" heartbeats to the inactive unit through any of its interfaces, even though one or more interfaces may be inoperative.

Changing the Cluster ID

Changing the cluster ID in a live environment is not recommended for two reasons. Firstly this will change the hardware address of the shared IPs and 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 would have to have their ARP caches flushed.

Secondly this breaks the connection between the security 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.

Invalid Checksums in Heartbeat Packets

Cluster Heartbeats packets are deliberately created with invalid checksums. This is done so that they will not be routed. Some routers may flag this invalid checksum in their log messages.

Making OSPF work

If OSPF is being used to determine routing metrics then a cluster **cannot** be used as the *designated router*.

If OSPF is to work then there must be another *designated router* available in the same *OSPF area* as the cluster. Ideally, there will also be a second, backup designated router to provide OSPF metrics should the main designated router fail.

12.5. Using Link Monitoring

When using an HA configuration, it can be important to use redundant paths to vital resources such as the Internet. The paths through the network from the master device in an HA configuration may fail in which case it may be desirable to have this failure trigger a failover to the slave unit which has a different path to the resource.

Monitoring the availability of specific network paths can be done with the CorePlus *Link Monitor*. The Link Monitor allows the administrator to specify particular hosts whose reachability is monitored using ICMP "Ping" requests and therefore link status. If these hosts become unreachable then the link is considered failed and a failover to a slave can be initiated. Provided that the slave is using a different network link and also monitoring the reachability of different hosts, traffic can continue to flow.

When using the Link Monitor in a High Availability cluster the option **Send from Shared IP Address** will be visible during Link Monitor setup. This option can be used if individual public IP addresses are not available for each unit in an HA cluster.

This feature is described further in Section 3.4.3, “The Link Monitor”.

12.6. HA Advanced Settings

The following CorePlus advanced settings are available for High Availability:

Sync Buffer Size

How much sync data, in Kbytes, to buffer while waiting for acknowledgments from the cluster peer.

Default: *1024*

Sync Packet Max Burst

The maximum number of state sync packets to send in a burst.

Default: *20*

Initial Silence

The time in seconds to stay silent on startup or after reconfiguration.

Default: *5*

Use Unique Shared Mac

Use a unique shared mac address for each interface. For further explanation of this setting see Section 12.3.5, “Using Unique Shared Mac Addresses”.

Default: *Enabled*

Deactivate Before Reconf

If enabled, this setting will make an active node failover to the inactive node before a reconfigure takes place instead of relying on the inactive node detecting that the active node is not operating normally and then taking over on its own initiative. Enabling this setting shortens the time where no node is active during configuration deployments.

Default: *Enabled*

Reconf Failover Time

Number of non-responsive seconds before failover at HA reconfiguration. The default value of zero means immediate reconfiguration.

Default: *0*

Chapter 13. Advanced Settings

This chapter describes the configurable advanced settings for CorePlus. The settings are divided up into the following categories:

Note: Activate after changes
After an advanced setting is changed an activate operation must be performed in order for the new CorePlus configuration to take effect.

13.1. IP Level Settings

Log Checksum Errors

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*

Log non IP4

Logs occurrences of IP packets that are not version 4. CorePlus only accepts version 4 IP packets; everything else is discarded.

Default: *Enabled*

Log Received TTL 0

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*

Block 0000 Src

Block 0.0.0.0 as source address.

Default: *Drop*

Block 0 Net

Block 0.* as source addresses.

Default: *DropLog*

Block 127 Net

Block 127.* as source addresses.

Default: *DropLog*

Block Multicast Src

Block multicast both source addresses (224.0.0.0 - 255.255.255.255).

Default: *DropLog*

TTL Min

The minimum TTL value accepted on receipt.

Default: *3*

TTL on Low

Determines the action taken on packets whose TTL falls below the stipulated TTLMin value.

Default: *DropLog*

Multicast TTL on Low

What action to take on too low multicast TTL values.

Default: *DropLog*

Default TTL

Indicates which TTL CorePlus is to use when originating a packet. These values are usually between 64 and 255.

Default: *255*

Layer Size Consistency

Verifies that the size information contained in each "layer" (Ethernet, IP, TCP, UDP, ICMP) is consistent with that of other layers.

Default: *ValidateLogBad*

SecuRemoteUDP Compatibility

Allow IP data to contain eight bytes more than the UDP total length field specifies. Checkpoint SecuRemote violates NAT-T drafts.

Default: *Disabled*

IP Option Sizes

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*

IP Option Source/Return

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*

IP Options Timestamps

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*

IP router alert option

How to handle IP packets with contained route alert.

Default: *ValidateLogBad*

IP Options Other

All options other than those specified above.

Default: *DropLog*

Directed Broadcasts

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*

IP Reserved Flag

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*

Strip DontFragment

Strip the Don’t Fragment flag for packets equal to or smaller than the size specified by this setting.

Default: *65535 bytes*

Multicast Mismatch option

What action to take when ethernet and IP multicast addresses does not match.

Default: *DropLog*

Min Broadcast TTL option

The shortest IP broadcast Time-To-Live value accepted on receipt.

Default: *1*

Low Broadcast TTL Action option

What action to take on too low broadcast TTL values.

Default: *DropLog*

13.2. TCP Level Settings

TCP Option Sizes

Verifies the size of TCP options. This function acts in the same way as IPOptionSizes described above.

Default: *ValidateLogBad*

TCP MSS Min

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*

TCP MSS on Low

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*

TCP MSS Max

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*

TCP MSS VPN Max

As is the case with TCPMSSMax, this is the highest Maximum Segment Size allowed. However, this setting 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*

TCP MSS On High

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*

TCP MSS Log Level

Determines when to log regarding too high TCP MSS, if not logged by TCPMSSOnHigh.

Default: *7000 bytes*

TCP Auto Clamping

Automatically clamp TCP MSS according to MTU of involved interfaces, in addition to TCPMSSMax.

Default: *Enabled*

TCP Zero Unused ACK

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*

TCP Zero Unused URG

Strips the URG pointers from all packets.

Default: *Enabled*

TCP Option WSOPT

Determines how CorePlus will handle window-scaling options. These are used to increase the size of the windows used by TCP; that is to say, 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*

TCP Option 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*

TCP Option 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*

TCP Option 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*

TCP Option 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*

TCP Option Con Timeout

Determines how CorePlus will handle connection count options.

Default: *StripLogBad*

TCP Option 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*

TCP SYN/URG

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*

TCP SYN/PSH

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*

TCP SYN/RST

The TCP RST flag together with SYN; normally invalid (strip=strip RST).

Default: *DropLog*

TCP SYN/FIN

The TCP FIN flag together with SYN; normally invalid (strip=strip FIN).

Default: *DropLog*

TCP FIN/URG

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*

TCP URG

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*

TCPE ECN

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*

TCP Reserved Field

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*

TCP NULL

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*

TCP Sequence Numbers

Determines if the sequence number range occupied by a TCP segment will be compared to the receive window announced by the receiving peer before the segment is forwarded.

TCP sequence number validation is only possible on connections tracked by the state-engine (not on packets forwarded using a **FwdFast** rule).

Possible values are:

Ignore - Do not validate. Means that sequence number validation is completely turned off.
ValidateSilent - Validate and pass on.
ValidateLogBad - Validate and pass on, log if bad.
ValidateReopen - Validate reopen attempt like normal traffic; validate and pass on.
ValidateReopenLog - Validate reopen attempts like normal traffic; validate, log if bad.
ReopenValidate - Do not validate reopen attempts at all; validate and pass on.
ReopenValidLog - Do not validate reopen attempts at all; validate, log if bad.

Default: *ValidateLogBad*

Notes on the *TCPSequenceNumbers* setting

The default *ValidateLogBad* (or the alternative *ValidateSilent*) will allow the de-facto behavior of TCP re-open attempts, meaning that they will reject re-open attempts with a previously used sequence number.

ValidateReopen and *ValidReopenLog* are special settings giving the default behavior found in older CorePlus versions where only re-open attempts using a sequence number falling inside the current (or last used) TCP window will be allowed. This is more restrictive than *ValidateLogBad*/*ValidateSilent*, and will block some valid TCP re-open attempts. The most significant impact of this will be that common web-surfing traffic (short but complete transactions requested from a relatively small set of clients, randomly occurring with an interval of a few seconds) will slow down considerably, while most "normal" TCP traffic will continue to work as usual.

Using either *ValidateReopen* or *ValidateReopenLog* is, however, not recommended since the same effect can be achieved by disallowing TCP re-open attempts altogether. These settings exist mostly for backwards compatibility.

ReopenValidate and *ReopenValidLog* are less restrictive variants than *ValidateLogBad* or *ValidateSilent*. Certain clients and/or operating systems might attempt to use a randomized sequence number when re-opening an old TCP connection (usually out of a concern for security) and this may not work well with these settings. Again, web-surfing traffic is most likely to be affected, although the impact is likely to occur randomly. Using these values instead of the default setting will completely disable sequence number validation for TCP re-open attempts. Once the connection has been established, normal TCP sequence number validation will be resumed.

Allow TCP Reopen

Allow clients to re-open TCP connections that are in the closed state.

Default: *Disabled*

13.3. ICMP Level Settings

ICMP Sends Per Sec Limit

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: *500*

Silently Drop State ICMPErrors

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 rule set for evaluation just like any other packet.

Default: *Enabled*

13.4. State Settings

Connection Replace

Allows new additions to CorePlus's connection list to replace the oldest connections if there is no available space.

Default: *ReplaceLog*

Log Open Fails

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*

Log Reverse Opens

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*

Log State Violations

Determines if CorePlus logs packets that violate the expected state switching diagram of a connection, for example, getting TCP FIN packets in response to TCP SYN packets.

Default: *Enabled*

Log Connections

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 SAT 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*

Log Connection Usage

This generates a log message for every packet that passes through a connection that is set up in the CorePlus state-engine. Traffic whose destination is the Clavister Security Gateway itself, for example CorePlus management traffic, is not subject to this setting.

The log message includes port, service, source/destination IP address and interface. This setting should only be enabled for diagnostic and testing purposes since it generates unwieldy volumes of log messages and can also significantly impair throughput performance.

Default: *Disabled*

Dynamic Max Connections

Allocate the Max Connection value dynamically.

Default: *Enabled*

Max Connections

This setting applies if **Dynamic Max Connections** above is disabled. Specifies how many connections CorePlus may keep open at any one time. Each connection consumes approximately 150 bytes RAM. When this setting is dynamic, CorePlus will try to use as many connections as is allowed by product.

Default: *8192*

13.5. Connection Timeout Settings

The settings in this section specify how long a connection can remain idle, that is to say with 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.

TCP SYN Idle Lifetime

Specifies in seconds how long a TCP connection, that is not yet fully established, is allowed to idle before being closed.

Default: *60*

TCP Idle Lifetime

Specifies in seconds how long a fully established TCP connection may idle before being closed. Connections become fully established once packets with their SYN flags off have travelled in both directions.

Default: *262144*

TCP FIN Idle Lifetime

Specifies in seconds 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*

UDP Idle Lifetime

Specifies in seconds how long UDP connections may idle before being closed. This timeout value is usually low, as UDP has no way of signalling when the connection is about to close.

Default: *130*

UDP Bidirectional Keep-alive

This allows both sides to keep a UDP connection alive. The default is for CorePlus to mark a connection as alive (not idle) every time data is sent from the side that opened the connection. Connections that do not receive any data from the opening side within the UDP lifetime will therefore be closed even if the other side continues to transmit data.

Default: *Disabled*

Ping Idle Lifetime

Specifies in seconds how long a Ping (ICMP ECHO) connection can remain idle before it is closed.

Default: *8*

IGMP Idle Lifetime

Connection lifetime for IGMP in seconds.

Default: *12*

Other Idle Lifetime

Specifies in seconds how long connections using an unknown protocol can remain idle before it is closed.

Default: *130*

13.6. Length Limit Settings

This section contains information about the size limits imposed on the protocols directly under IP level, such as TCP, UDP and ICMP.

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.

Max TCP Length

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*

Max UDP Length

Specifies in bytes 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*

Max ICMP Length

Specifies in bytes 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*

Max GRE Length

Specifies in bytes 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*

Max ESP Length

Specifies in bytes 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*

Max AH Length

Specifies in bytes 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*

Max SKIP Length

Specifies in bytes the maximum size of a SKIP packet.

Default: *2000*

Max OSPF Length

Specifies the maximum size of an OSPF packet. OSPF is a routing protocol mainly used in larger LANs.

Default: *1480*

Max IPIP/FWZ Length

Specifies in bytes 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*

Max IPsec IPComp Length

Specifies in bytes the maximum size of an IPComp packet.

Default: *2000*

Max L2TP Length

Specifies in bytes the maximum size of a Layer 2 Tunneling Protocol packet.

Default: *2000*

Max Other Length

Specifies in bytes the maximum size of packets belonging to protocols that are not specified above.

Default: *1480*

Log Oversized Packets

Specifies if CorePlus will log occurrences of oversized packets.

Default: *Enabled*

13.7. 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.

Many IP stacks, however, 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.

Pseudo Reass Max Concurrent

Maximum number of concurrent fragment reassemblies. To drop all fragmented packets, set PseudoReass_MaxConcurrent to 0.

Default: *1024*

Illegal Fragments

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".*

Duplicated Fragment Data

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*

Failed Fragment Reassembly

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 IllegalFrags 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*

Dropped Fragments

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 DroppedFrags 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 rule set.
- *LogSuspect* - Logs individual dropped fragments of reassembly attempts affected by "suspect" fragments.
- *LogAll* - Always logs individual dropped fragments.

Default: *LogSuspect*

Duplicate Fragments

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. DuplicateFrags 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*

Fragmented ICMP

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*

Minimum Fragment Length

Minimum Fragment Length determines how small all fragments, with the exception of the final fragment, of a packet can be expressed in bytes.

Although the arrival of too many fragments that are too small may cause problems for IP stacks, it is usually not possible to set this limit too high. It is rarely the case that senders create very small fragments. However, a sender may send 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*

Reassembly Timeout

A reassembly attempt will be interrupted if no further fragments arrive within Reassembly Timeout seconds of receipt of the previous fragment.

Default: *65*

Max Reassembly Time Limit

A reassembly attempt will always be interrupted Reassembly Time Limit seconds after the first received fragment arrived.

Default: *90*

Reassembly Done Limit

Once a packet has been reassembled, CorePlus is able to remember reassembly for this number of seconds in order to prevent further fragments, for example old duplicate fragments, of that packet from arriving.

Default: *20*

Reassembly Illegal Limit

Once a whole packet has been marked as illegal, CorePlus is able to retain this in memory for this number of seconds in order to prevent further fragments of that packet from arriving.

Default: *60*

13.8. Local Fragment Reassembly Settings

Max Concurrent

Maximum number of concurrent local reassemblies.

Default: *256*

Max Size

Maximum size of a locally reassembled packet.

Default: *10000*

Large Buffers

Number of large (over 2K) local reassembly buffers (of the above size).

Default: *32*

13.9. SSL Settings

SSL Processing Priority

The amount of of CPU time that SSL processing is allowed to use.

Default: *Normal*

TLS RSA 3DES 168 SHA1

Enable cipher RSA_WITH_3DES_168_SHA1.

Default: *Enabled*

TLS RSA RC4 128 SHA1

Enable cipher RSA_WITH_RC4_128_SHA1.

Default: *Enabled*

TLS RSA RC4 128 MD5

Enable cipher TLS_RSA_WITH_RC4_128_MD5.

Default: *Enabled*

TLS RSA EXPORT 1024 RC4 56 SHA1

Enable cipher TLS_RSA_EXPORT1024_WITH_RC4_56_SHA1.

Default: *Enabled*

TLS RSA EXPORT 1024 RC4 40 MD5

Enable cipher TLS_RSA_EXPORT1024_WITH_RC4_40_MD5.

Default: *Disabled*

TLS RSA EXPORT 1024 RC2 40 MD5

Enable cipher TLS_RSA_EXPORT1024_WITH_RC2_40_MD5.

Default: *Disabled*

TLS RSA EXPORT NULL SHA1

Enable cipher TLS_RSA_EXPORT_WITH_NULL_SHA1 (no encryption, just message validation).

Default: *Disabled*

TLS RSA EXPORT NULL MD5

Enable cipher TLS_RSA_EXPORT_WITH_NULL_MD5 (no encryption, just message validation).

Default: *Disabled*

13.10. Miscellaneous Settings

UDP Source Port 0

How to treat UDP packets with source port 0.

Default: *DropLog*

Port 0

How to treat TCP/UDP packets with destination port 0 and TCP packets with source port 0.

Default: *DropLog*

Watchdog Time

Number of non-responsive seconds before watchdog is triggered (0=disable).

Default: *180*

Flood Reboot Time

As a final way out, CorePlus automatically reboots if its buffers have been flooded for a long time. This setting specifies this amount of time.

Default: *3600*

Dynamic High Buffers

This setting decides if CorePlus will automatically specify the High Buffers value. The default value is *3% of total available memory, with a lower limit of 1024*
Note that in addition to this there is always an extra 512 Kbytes allocated for buffers.

This setting requires a hardware restart for a new value to take effect. A reconfiguration is not sufficient.

Default: *Enabled*

High Buffers

If **Dynamic High Buffers** is not enabled then this number of buffers will be allocated in RAM above the 1 MByte limit. When troubleshooting problems raising this value can sometimes provide a solution.

This setting requires a hardware restart for a new value to take effect. A reconfiguration is not sufficient.

Default: *1024*

Max Connections

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 subsystems of such overlaps. The associated settings limit memory used by the re-assembly subsystem.

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*

Max Memory

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*

Timeout

The time in seconds before CorePlus automatically enables its screen saver for a CorePlus Software only installation. 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)*

Screen Saver Selection

The type of screen saver used.

Default: *Blank*

Status Bar Selection

The status bar control.

Default: *Auto*

Max Pipe Users

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 setting. For more information about pipes and pipe users, see Section 11.1, “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 Clavister Customer Web 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 server addresses are set correctly 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 the following ways:

- 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 subsscription is renewed well before the current subscription ends.

IDP and Anti-Virus Database Updating

These service operate by downloading "signature" patterns which are used by CorePlus to search for

the most recently recognized 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 needs 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 explicitly set. 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:

```
Device:/>  updatecenter -update idp
```

An Anti-Virus update can similarly be initiated with the command:

```
Device:/>  updatecenter -update av
```

Querying Update Status

To get the status of IDP updates use the command:

```
Device:/>  updatecenter -status idp
```

To get the status of AV updates:

```
Device:/>  updatecenter -status av
```

Querying Server Status

To get the status of the Clavister network servers use the command:

```
Device:/>  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

```
Device:/>  removedb idp
```

To remove the Anti-Virus database, use the command:

```
Device:/>  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: A pause can occur when updating

Anti-Virus database updates may require a couple of seconds to be optimized once an update is downloaded. This will cause the Clavister Security Gateway to sometimes momentarily pause in its operation. This behavior is normal. Deleting the databases can give rise to a similar pause.

Appendix B. 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 7.5, “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_INFECTION	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_OPENSSH	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

Group Name	Intrusion Type
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 C. Verified MIME filetypes

Some CorePlus Application Layer Gateways (ALGs) have the optional ability to verify that the contents of a downloaded file matches the type that the filetype in the filename indicates. The filetypes for which MIME verification can be done are listed in this appendix and the ALGs to which this applies are:

- The HTTP ALG
- The FTP ALG
- The POP3 ALG
- The SMTP ALG

The ALGs listed above also offer the option to explicitly allow or block certain filetypes as downloads from a list of types. That list is the same one found in this appendix.

For a more detailed description of MIME verification and the filetype block/allow feature, see Section 7.2.2, “The HTTP ALG”.

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
avr	Audio Visual Research Sound
aw	Applix Word file
bh	Blackhole archive format file
bmp	Windows Bitmap Graphics
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

Filetype extension	Application
cpl	Windows Control Panel Extension file
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
eet	EET archive
egg	Allegro datafile
elc	eMacs 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
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
lha	LHA compressed archive file
lim	Limit compressed archive
lisp	LIM archive data
lzh	LZH compressed archive file
md	MDCD compressed archive file
mdb	Microsoft Access Database
mid,midi	Musical Instrument Digital Interface MIDI-sequence Sound
mmf	Yamaha SMAF Synthetic Music Mobile Application Format
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

Filetype extension	Application
mpv	MPEG-1 Video file
Microsoft files	Microsoft office files, and other Microsoft files
msa	Atari MSA archive data
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
pbf	Portable Bitmap Format Image
pbm	Portable Bitmap Graphic
pdf	Acrobat Portable Document Format
pe	Portable Executable file
pfb	PostScript Type 1 Font
pgm	Portable Graymap Graphic
pkg	SysV R4 PKG Datastreams
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
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
svr4	SysV R4 PKG Datastreams
swf	Macromedia Flash Format file
tar	Tape archive file

Filetype extension	Application
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
z	Unix compressed file

Appendix D. The OSI Framework

Overview

The Open Systems Interconnection Model defines a framework for inter-computer 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. The model is relevant to understanding the operation of many CorePlus features such as ARP, Services and ALGs.

Figure D.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

Layer Functions

The different layers perform the following functions:

Layer 7 - Application Layer — Defines the user interface that supports applications directly. Protocols: HTTP, FTP, TFTP. DNS, SMTP, Telnet, SNMP and similar. The ALGs operate at this level.

Layer 6 - Presentation Layer — Translates the various applications to uniform network formats that the rest of the layers can understand.

Layer 5 - Session Layer — Establishes, maintains and terminates sessions across the network. Protocols: NetBIOS, RPC and similar.

Layer 4 - Transport Layer — Controls data flow and provides error-handling. Protocols: TCP, UDP and similar.

Layer 3 - Network Layer — Performs addressing and routing. Protocols: IP, OSPF, ICMP, IGMP and similar.

Layer 2 - Data-Link Layer — Creates frames of data for transmission over the physical layer and includes error checking/correction. Protocols: Ethernet, PPP and similar. ARP operates at this level.

Layer 1 - Physical Layer — Defines the physical hardware connection.

Alphabetical Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X