The Clavister

cOS Core Cookbook

2nd edition

Just the tastiest recipes

by Peter Nilsson
Issued by
Clavister AB Sweden

Printed by CreateSpace

Second Printing, 2017

ISBN: 978-91-982968-6-0

Clavister AB
Sjögatan 6J
SE-89160 Örnsköldsvik
Sweden

www.clavister.com

cookbook@clavister.com

Acknowledgements

A huge thank you for the encouragement and support of the following people in helping to get this book finished: Colin Mummery, Neal Sutherland, Kimmo Klemola, Tobias Rhén, Torbjörn Fahlen, Jaap van Duijn, André Sjöström, Fredrick Backman and Edgard Dias Batista.

About the Author

Peter Nilsson is a senior support engineer with Clavister AB and has over a decade of experience helping enterprises set up network security solutions using Clavister products. He lives in Örnsköldsvik, Sweden.

Table of Contents

Chapter 1: Fundamentals

The purpose of this book is to introduce a new user to the cOS Core network operating system. cOS Core is the software that powers Clavister's principal product range of Next Generation Firewalls. cOS Core provides a wide range of features for a network administrator to control, restrict and monitor the data traffic passing through a Clavister firewall. That traffic flow could be between any types of networks, either public or private.

cOS Core does not run as software on top of another operating system, such as Linux. Instead, it runs directly on its hardware platform and is therefore, itself, a true operating system. The absence of an underlying host operating system means that cOS Core combines high performance with a small memory footprint and is well suited to environments with hardware resource constraints. cOS Core can run both on Clavister hardware or under a hypervisor in a virtual environment.

This chapter will describe the fundamental principles that cOS Core uses regarding IP policies, management, routing, routing principles and other functions. If you are already familiar with these principles, you can skip to the next chapter.

The separate *Clavister cOS Core Administration Guide* describes each feature and function in greater detail. This cookbook will concentrate more on solutions and scenarios rather than try to explain in depth what every setting does. It is recommended to combine this cookbook with the cOS Core Administration Guide in order to get a more in-depth understanding of cOS Core. The Administration Guide is almost a thousand pages long and it is included as part of the PDF file documentation set for every cOS Core release. It can be downloaded from Clavister's corporate website (www.clavister.com).

1.1. The Concept of Objects

This section will explain the concept and usage of cOS Core *Objects*. It will also talk generally about using the *Web User Interface* (WebUI) for managing cOS Core (any web browser can be used to connect to cOS Core for management). How to use the WebUI, when not to use it, and why it's important to keep everything well structured right from the start.

There are other ways to manage cOS Core as well, such as via SSH, InControl or by using the console port. In this book we will mainly be using the WebUI for management.

Objects can be used in all aspects of a cOS Core configuration. For instance, we can create an IP object called "My_IP" and give it an IP address such as 192.168.1.1. We can then use that object's name in other places such as with IP Policies, Interfaces or User Authentication.

Using address book objects has some important benefits:

- It increases understanding of the configuration by using meaningful symbolic names.
- Using address object names, instead of entering numerical addresses, reduces errors.
- If changes are needed on an address book object, all functions and features that use that object will be updated automatically.

We will go into more detail on how to use the various objects as we progress through the book.

Using the Address Book With Other Objects

There are many types of cOS Core configuration objects that can be created. In order to keep to the basics, for now we will only discuss IP address objects in the Address Book as well as Service objects. Other object types will be explained in later recipes as we progress.

Figure 1.1.1 is a screenshot from the management web browser interface (WebUI) that shows the first part of the **Objects** menu.

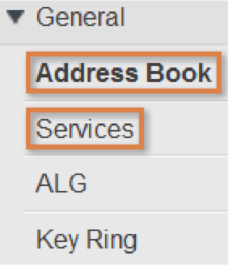
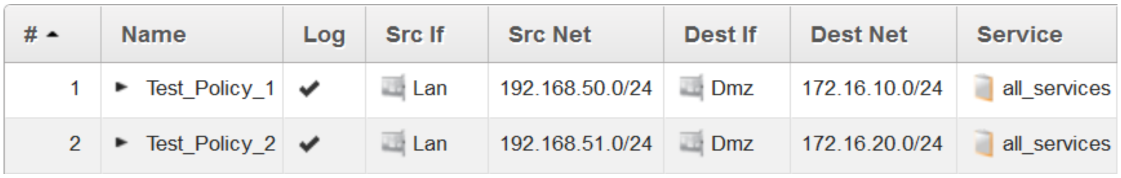


Figure 1.1.1 Object view

When we create, for example, an IP Policy, we get the option to either manually input a numeric IP address or network directly, or we can use a named object that already exists in the address book.

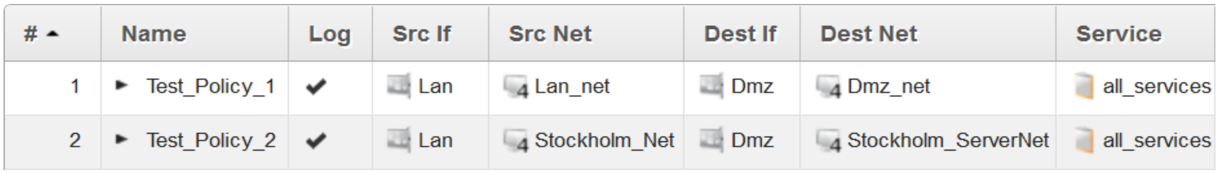
It is highly recommended to use addresses from the address book with IP policies, routing and other object types. The configuration will become difficult to read and structure if we enter numeric IP addresses and networks directly. For example, the IP Policies shown in the WebUI screenshot below use numeric IP addresses instead of address book objects.



#	Name	Log	Src If	Src Net	Dest If	Dest Net	Service
1	Test_Policy_1	✓	Lan	192.168.50.0/24	Dmz	172.16.10.0/24	all_services
2	Test_Policy_2	✓	Lan	192.168.51.0/24	Dmz	172.16.20.0/24	all_services

Figure 1.1.2 IP Policies without address book objects

Our IP policies, or any other part of the configuration, will be better structured and easier to read if we use address book objects. This is shown in the figure below where the IP Policies from above now use address book objects instead of numeric values.



#	Name	Log	Src If	Src Net	Dest If	Dest Net	Service
1	Test_Policy_1	✓	Lan	Lan_net	Dmz	Dmz_net	all_services
2	Test_Policy_2	✓	Lan	Stockholm_Net	Dmz	Stockholm_ServerNet	all_services

Figure 1.1.3 IP Policies with address book objects

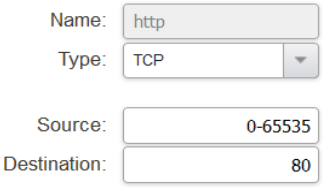
However, it is still possible to mouse-over an address book object to get a tool-tip which shows the numeric IP address or group membership.

The greatest advantage of using address book objects is when we need to change an existing address book object. Changing the object in the address book means that all locations in a cOS Core configuration where the object is used will be updated automatically. There is therefore no need to go into each individual policy, route, user authentication and so on, manually and update each one to reflect the new object value.

Services

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, which is associated with a specific source and/or destination port number(s). For example, the HTTP service is defined as using the TCP protocol with the associated destination port 80 and any source port.

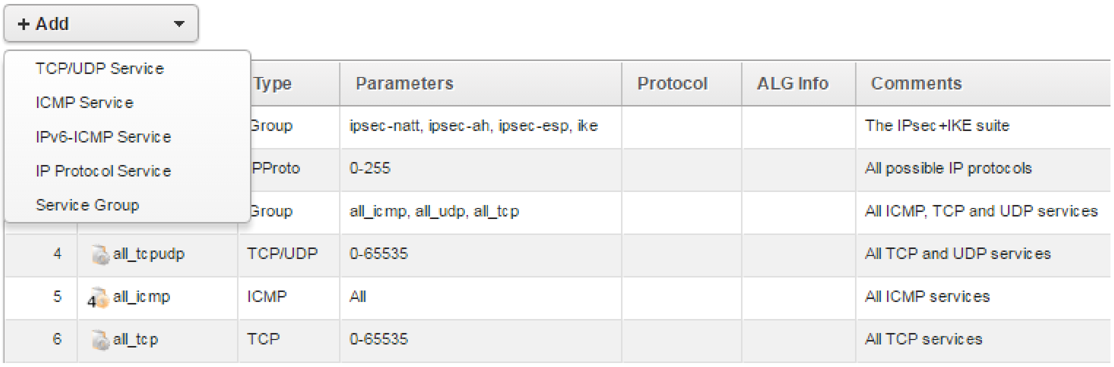
The screenshot below shows the basic properties of a cOS Core Service object.



Name: http
Type: TCP
Source: 0-65535
Destination: 80

Figure 1.1.4 Service object properties

However, service objects are not restricted to just the TCP or UDP protocols. They can be used to include ICMP messages as well as user-definable IP protocols, as shown in figure 1.1.5.



		Type	Parameters	Protocol	ALG Info	Comments
		Group	ipsec-natt, ipsec-ah, ipsec-esp, ike			The IPsec+IKE suite
		PProto	0-255			All possible IP protocols
		Group	all_icmp, all_udp, all_tcp			All ICMP, TCP and UDP services
4	all_tcpudp	TCP/UDP	0-65535			All TCP and UDP services
5	all_icmp	ICMP	All			All ICMP services
6	all_tcp	TCP	0-65535			All TCP services

Figure 1.1.5 Service object types

1.2. Introduction to IP Policies

A fundamental function of cOS Core is found in IP policies (sometimes referred to as security policies). This section will describe what an IP policy is and how to use it, along with some tips.

Differences Between an IP Policy and an IP Rule

There are two main types of traffic rules: IP Policies and IP Rules. Adding either of these can be seen in the WebUI screenshot below.

We will go through some of the other types of traffic rules seen in the screenshot as we progress through the book.

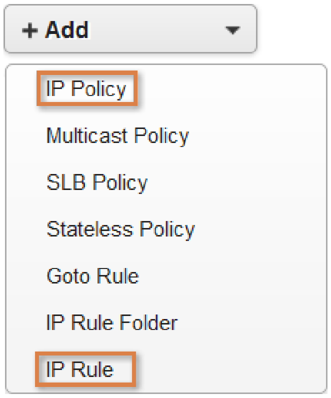


Figure 1.2.1 Adding a new IP Policy or IP Rule

IP Policies and IP Rules are two types of rules that do the same thing. The purpose of IP policies is to help users create rules by reducing the required number of rules and steps by automatically creating additional rules when needed (depending on the action type and networks used).

We will base all of our examples, descriptions and illustrations in this book on IP Policies instead of IP Rules as they fill the same purpose but IP Policies are easier to work with and supports some features that IP rules do not, such as Email control for IMAP.

Since we encourage our users to use IP Policies instead of IP Rules, IP Rules are disabled by default in new configurations. They can be enabled again by going to Advanced settings under the System tab and then check the Allow IPRules box under Misc. Settings.

IP Policy Properties

When creating an IP policy, a number of different fields can be modified.

These fields defines how cOS Core should treat the IP policy in question. For example, should it hide the client's source IP/network by using Address Translation? Or perhaps it should drop traffic without telling the client (Deny)?

The most common IP policy properties are the following:

- **Action**

  With the action we can decide whether we should Allow or Deny the traffic from the defined source interface + network to the destination interface + network.

- **Deny Behavior**

  If you choose the Action Deny you will get an extra option called Deny Behavior. The Drop behavior will drop the packet from the defined source interface + network to the destination interface + network without telling the client that the packet was dropped. The Reject behavior can be used if the client should be informed that the target service is unavailable. From a security perspective, it is recommended to use Drop instead of Reject as it is usually better to say nothing at all instead of telling the client that there may be something there.

- **Address Translation**

  Address translation can be used to either change the source IP that we will use when sending traffic, or to change which destination IP the traffic should be sent towards.

- **NAT**(Network Address Translation)

  NAT can only be configured on Source Address Translation. A NAT address translation means that we mask the source IP of the traffic. This translation is normally used when clients on private IP addresses want to reach the Internet and their source IP addresses must be changed before leaving cOS Core.

- **SAT** (Static Address Translation)

  SAT can be configured on Destination as well as Source Address Translation. A SAT address translation means we translate the destination or source IP (depending on what we choose) to a specific IP, set in the New IP Address field. This kind of translation is normally used to give external access to internal resources, such as a web server.

- **Auto**

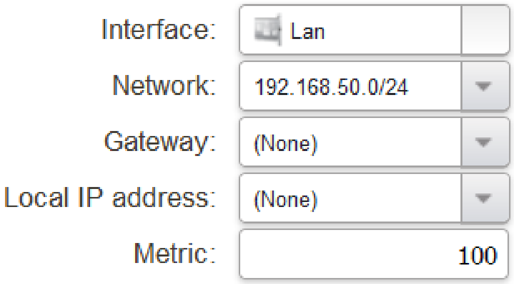
  Automatic address translation can only be configured on Source Address Translation. Auto translation will check whether the traffic should be NAT'ed or not depending on the source and destination. If both the source IP and destination IP are classified as private IP addresses, NAT will not be used. If the source IP is classified as private and the destination IP is classified as a public IP, NAT will be used.

We will explain other features of the IP Policy as we progress through the book.

1.3. Introduction to Routing

IP routing is one of the most fundamental functions of cOS Core. Any IP packet flowing through a Clavister Next Generation Firewall will be subjected to at least one routing decision at some point in time, and properly configured routes are crucial for the system to function as expected.

A *Route* defines where a network is located. Let's look at the simplest definition of a route by choosing an interface and a network such as the one shown in the next screenshot.



Interface: Lan

Network: 192.168.50.0/24

Gateway: (None)

Local IP address: (None)

Metric: 100

Figure 1.3.1 A basic route

Note

It is not recommended to enter a network address directly, but we will do it in this example so it is easier to understand. Usually, an address book object would be used.

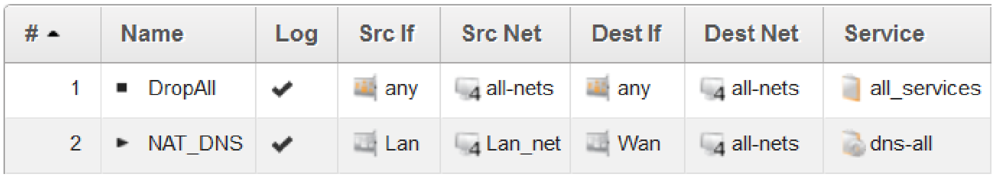
In the previous screenshot, we are telling cOS Core that in order to find hosts in the 192.168.50.0/24 network, we must use the LAN interface. When configuring routes, configure them as if cOS Core is asking the following question: "I want to reach network XXXX. Behind which interface can I find it?"

1.4. Two Important cOS Core Principles

Principle 1: Rule Ordering

By rules, we mean IP policies/rules, User Authentication Rules, Remote Management Rules, Pipe Rules and more. Basically, anything related to configurable rules that determine how a function or feature behaves. The main principle of such rules is that they are read from the top to the bottom. This means that cOS Core will traverse the specific rule list/set until it finds a matching rule and when a match has been found the rule processing stops.

This makes it extremely important that our rules are configured in the correct order. To make a simple IP policy example, we have two policies that look like this:

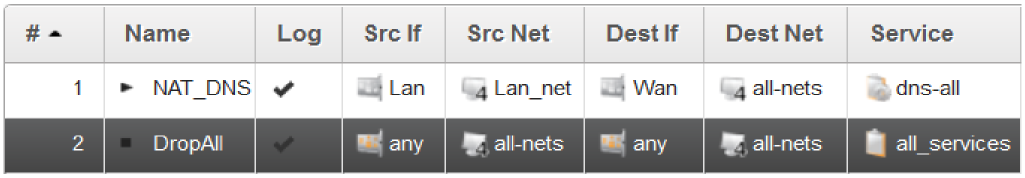


# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service
1	■ DropAll	✓	any	all-nets	any	all-nets	all_services
2	► NAT_DNS	✓	Lan	Lan_net	Wan	all-nets	dns-all

Figure 1.4.1 Problematic IP policy ordering

In this case, we have a DropAll policy at position #1 and a new policy at position #2 that allows DNS queries from the LAN interface and network. The second policy #2 in this scenario will never trigger because the policy above it matches first and drops the traffic. Since a matching policy has been found, no further rule processing will be performed.

By moving our new DNS policy above the DropAll policy, as shown in the next screenshot, we correct the problem.

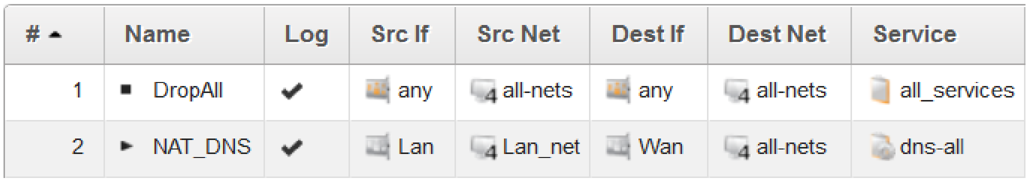


# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service
1	► NAT_DNS	✓	Lan	Lan_net	Wan	all-nets	dns-all
2	■ DropAll	✓	any	all-nets	any	all-nets	all_services

Figure 1.4.2 Correct IP policy ordering

Detecting IP Rule Set Problems

Understanding how to read the rules is very important because otherwise it can inadvertently open our network for both incoming and outgoing connections. If we look at the DropAll policy in the next screenshot and read it from the left to the right, we will have the following matches occurring:



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service
1	■ DropAll	✓	any	all-nets	any	all-nets	all_services
2	► NAT_DNS	✓	Lan	Lan_net	Wan	all-nets	dns-all

Figure 1.4.3 Problematic IP policy ordering

- Source interface is set to **any**. Match? Yes.
- Source network is set to **all-nets**. Match? Yes.
- Destination interface is set to **any**. Match? Yes.
- Destination network is set to **all-nets**. Match? Yes.
- Network service is set to **all_services**. Match? Yes.

So, all the criteria for a policy match has been met and the policy will trigger, and in this case drop the traffic. cOS Core will not continue to search for any other policy matches as it has found what it was looking for based on where the traffic was received, where it was going and what port (or protocol) was used.

Principle 2: Routing

When a routing table is evaluated, the ordering of the routes is not important. Instead, all routes in the relevant routing table are evaluated and the "smallest" route is used. To explain what "smallest" route means, let's consider if we have the following two routes:

1. *Interface=LAN Network=10.10.10.0/24*

2. *Interface=DMZ Network=10.10.10.0/16*

Since the first route has a /24 network of 256 addresses and the second route has a /16 network of 65536 addresses, the first route is considered smaller and can give a match first.

A very common log entry related to routing problems is a log event containing the text "Default_Access_Rule". This will occur in the above scenario if traffic from IP address 10.10.10.10 is received on the DMZ (DeMilitarized Zone) interface, because the LAN network (/24) is smaller than the DMZ network (/16) and the smaller network will take precedence.

If traffic is appearing on an interface where cOS Core is not expecting it, the traffic will also be dropped and this will generate a "Default_Access_Rule" log event message.

Using Route Metrics

The *Route Metric* can be used to tell cOS Core which route it should use in case there are several routes that are identical. The metric value is set as a parameter on a route. For example:

1. *Interface=Wan1 Network=all-nets Gateway=ISP1-GW Metric=50*

2. *Interface=Wan2 Network=all-nets Gateway=ISP2-GW Metric=100*

In this example, the interfaces are different but the network is the same. By setting a lower metric on the primary route for interface Wan1, we tell cOS Core that in the case of a network conflict, it should use this route instead of the Wan2 interface route.

Note

An identical route with the same metric can be a normal occurrence when used in conjunction with server load balancing. Server load balancing is discussed in a later chapter.

Identical Routes With Same Metric

Another common scenario that can occur is a situation where both the route and the metric are identical, such as in this example:

1. *Interface=Wan1 Network=all-nets Gateway=ISP1-GW Metric=100*

2. *Interface=Wan2 Network=all-nets Gateway=ISP2-GW Metric=100*

In this example, cOS Core will be unable to determine which of the two routes it should use. Their network size is the same and the metrics are also the same. The administrator wants cOS Core to use Wan1 as its primary Internet provider interface and this particular setup could work just fine for months, then all of a sudden it could stop working.

The reason for this is that since cOS Core will be unable to determine which route it should use, it will use a random one. After a system restart it may choose a different route, so we could get the false impression that it works and that things are configured correctly.

In order to address this problem we need to modify the metric on either the primary or secondary route. If we want Wan1 to be the primary interface towards the Internet provider (in this scenario) we need to set its metric to be lower than the Wan2 interface route, like this:

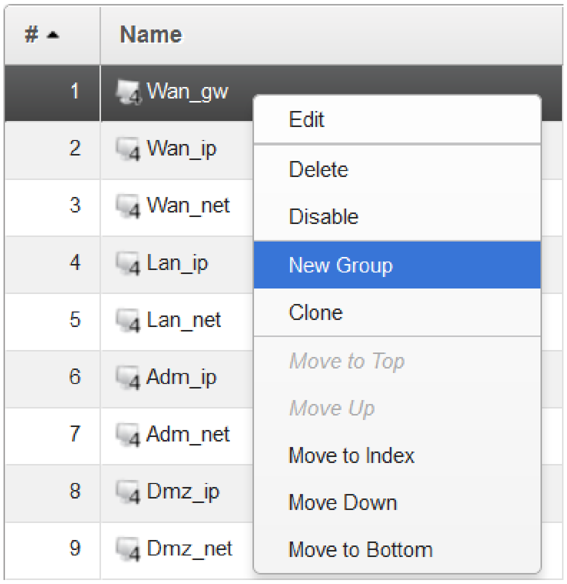
1. *Interface=Wan1 Network=all-nets Gateway=ISP1-GW Metric=50*

2. *Interface=Wan2 Network=all-nets Gateway=ISP2-GW Metric=100*

1.5. Using Comment Groups

Using comment groups to distinguish objects and rules is highly recommended. Using groups with different colors and grouping will help significantly when reading larger configurations. Performing comment group structuring of the address book, rules, routes and so on, as early as possible is recommended.

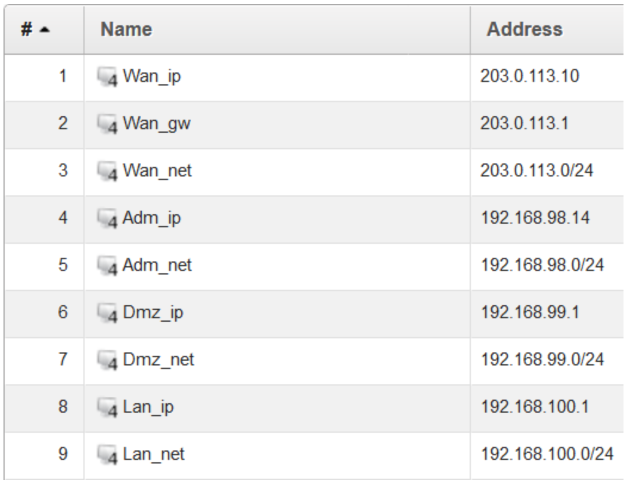
To add a new comment group, select the first object we want to be part of the group, right-click and select **New Group**. The example in the next screenshot shows this being done on the address book.



# ▲	Name
1	Wan_gw
2	Wan_ip
3	Wan_net
4	Lan_ip
5	Lan_net
6	Adm_ip
7	Adm_net
8	Dmz_ip
9	Dmz_net

Figure 1.5.1 Adding a new comment group

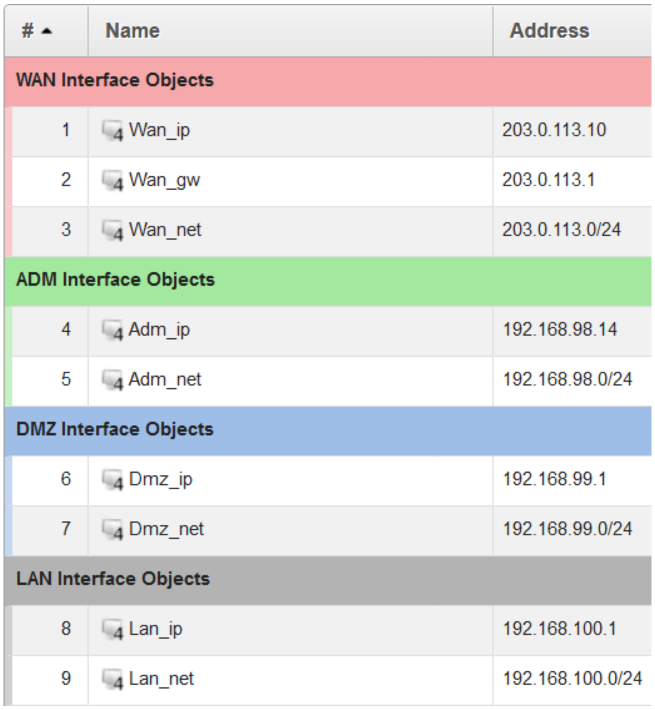
Before adding comment groups, the address book might look something like the address book shown in the screenshot below.



# ▲	Name	Address
1	Wan_ip	203.0.113.10
2	Wan_gw	203.0.113.1
3	Wan_net	203.0.113.0/24
4	Adm_ip	192.168.98.14
5	Adm_net	192.168.98.0/24
6	Dmz_ip	192.168.99.1
7	Dmz_net	192.168.99.0/24
8	Lan_ip	192.168.100.1
9	Lan_net	192.168.100.0/24

Figure 1.5.2 Address book without comment group separation

When all the objects that are part of a group are distinguished by a different color, the address book becomes much easier to read.



# ▲	Name	Address
WAN Interface Objects		
1	Wan_ip	203.0.113.10
2	Wan_gw	203.0.113.1
3	Wan_net	203.0.113.0/24
ADM Interface Objects		
4	Adm_ip	192.168.98.14
5	Adm_net	192.168.98.0/24
DMZ Interface Objects		
6	Dmz_ip	192.168.99.1
7	Dmz_net	192.168.99.0/24
LAN Interface Objects		
8	Lan_ip	192.168.100.1
9	Lan_net	192.168.100.0/24

Figure 1.5.3 Address book with comment group separation

Chapter 2: Basic Setup

This chapter will describe how to perform the basic configuration of cOS Core in order to achieve Internet access along with configuring basic routing and IP policies.

The objective of this chapter is to get started with configuring cOS Core and become familiar with the Web User Interface (WebUI) and various other aspects of the basic setup procedure such as remote management rules, licensing, firmware updates, allowing outgoing connections and more.

The Clavister Next Generation Firewall used in this solution will have 4 physical interfaces which have the logical names WAN, LAN, DMZ and ADM.

Network Diagram Icons

As we move through the recipes in the book, we will encounter network diagrams that illustrate the particular scenario we are looking at.

The network diagrams use a consistent set of icons which are shown below in *Figure 2.1*, along with a short description of what they represent.

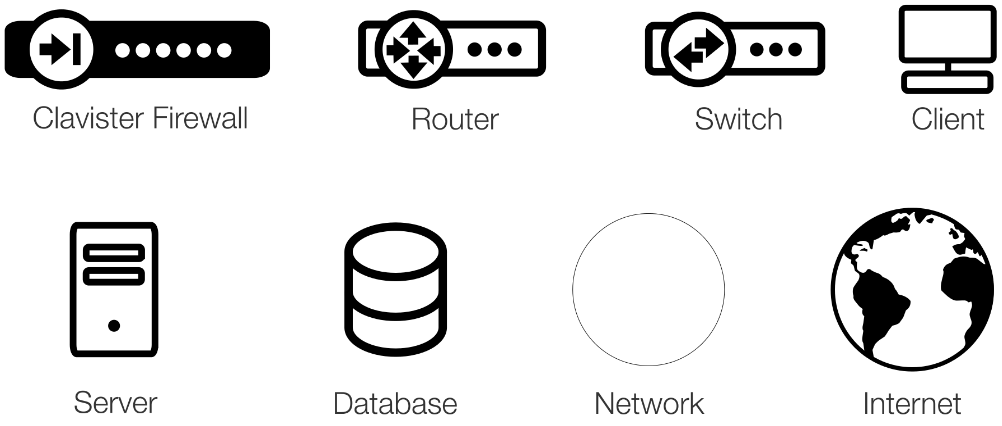


Figure 2.1 Cookbook Diagram Icons

Recipe 2.1. Basic installation, licensing, backup and firmware updates

Objectives

The objective of this recipe is to get management access to the Clavister Next Generation Firewall to review the remote management rules, install a license on the firewall and to perform a firmware update.

Detailed Discussion

When booting up the system for the first time, cOS Core automatically makes management access available on a single predefined Ethernet interface and assigns the private IPv4 address 192.168.1.1 to it. The first interface is normally used for the management workstation connection.

This management interface is labeled **If1** in the complete interface list shown next in *Table 2.1.1*.

Interface Name	IP Address
core	127.0.0.1
If1	192.168.1.1
If2	127.0.2.1
If3	127.0.3.1
If4	127.0.4.1

Table 2.1.1 The default management interface

Connecting to the cOS Core can be done using any of the following methods:

- Network connection to the WebUI using HTTP (TCP Port 80).
- Network connection to the WebUI using encrypted HTTPS (TCP port 443).

- Network connection to the cOS Core CLI using SSH (TCP port 22).
- Connecting to the cOS Core CLI directly via the firewall's Console Port.

> Note
>
> *Centralized management configuration and access will not be discussed or used in this book.*

Changing the default management IP and network using the CLI

The default IP address and network set on the management interface may not always match what the administrator wants. If the connecting client is not part of the 192.168.1.0/24 network and connected to the first interface, WebUI and SSH access is not possible until this has been changed.

The only way to change this is to use CLI via the local console. We will not go into details about CLI usage and functionality as the majority of the book will be based on the WebUI.

Since it is a common question, we will list below the two basic CLI commands needed to change the default IP address and network on the management interface and also to update the remote management rule(s) to allow access from our new network.

```
Device:/> set Interface Ethernet if1
                       IP=192.168.98.14
                       Network=192.168.98.0/24

Device:/> set RemoteManagement RemoteMgmtHTTP rmgmt_http
                       Network=192.168.98.0/24
                       Interface=if1
```

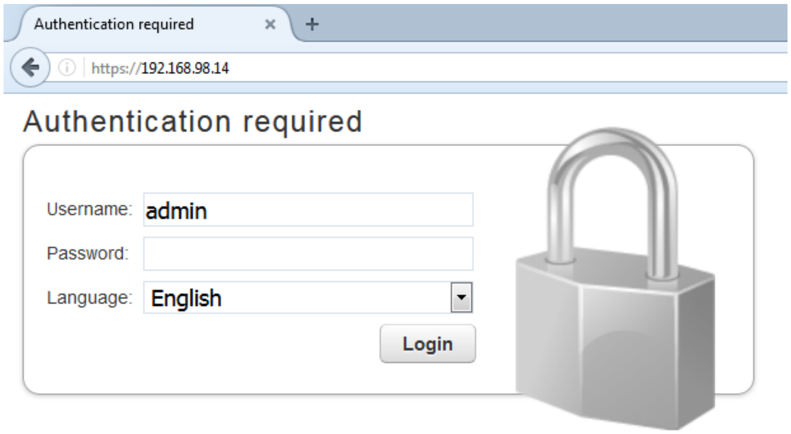
For more information about the CLI, please see the cOS Core CLI Reference Guide.

Note

With the default configuration the WebUI is only accessible via HTTPS since HTTP management is disabled by default.

Connecting to the WebUI

To connect to the WebUI, open a browser such as Firefox, Chrome or Opera, and go to the IP address of the management interface using HTTPS. In our setup it will be https://192.168.98.14, as shown in the next screenshot.



Authentication required

Username: admin

Password:

Language: English

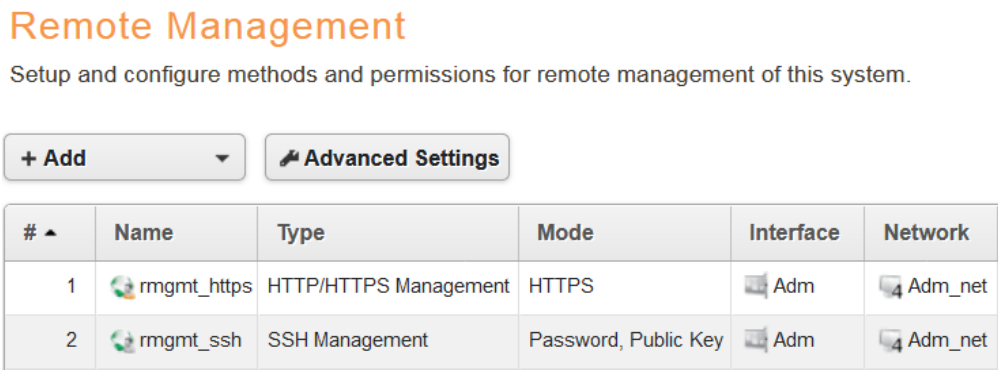
Figure 2.1.2 The cOS Core login dialog

The default login and password for a new installation are both "**admin**".

The remote management rules

Once we have access to the WebUI, it will be a good idea to review the current management access rules. These rules control the access levels to the WebUI, SSH, SNMP, REST API and InControl, and from which interface(s) and network(s) clients are allowed to connect in order to manage cOS Core.

The remote management rules can be found in the WebUI under **System > Remote Management**.



Remote Management

Setup and configure methods and permissions for remote management of this system.

# ▲	Name	Type	Mode	Interface	Network
1	rmgmt_https	HTTP/HTTPS Management	HTTPS	Adm	Adm_net
2	rmgmt_ssh	SSH Management	Password, Public Key	Adm	Adm_net

Figure 2.1.3 Remote management rules

In the rules shown above, we have configured management access using HTTPS (rule **1**) and SSH (rule **2**) from the Adm interface and the Adm_ net network, which corresponds to 192.168.98.0/24 in the address book. With these remote management rules we have now restricted access, the only way to be able to connect and reach the management WebUI (and use SSH) now is by being part of the Adm_net network and to be located behind the ADM interface.

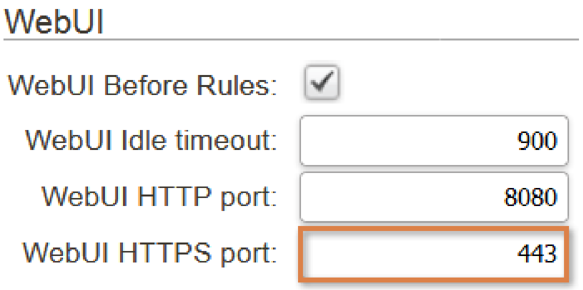
If we attempt to connect to the WebUI from any other interface or network, access will be denied. The client will not be presented with a login prompt as cOS Core will actively drop the connection attempt without reporting anything back to the client, as the user does not have remote management access.

Further restricting management access

In many scenarios, the firewall will be placed in key locations in the network, so it is essential to make sure that no unauthorized access is allowed. To further strengthen security we are going to do two things:

Change the HTTPS port to something else.

To change the HTTPS port, go to **System > Remote Management > Advanced Settings** and change the HTTPS Port to be the desired port as shown below. We will leave it on default port 443 in this chapter, as seen in Figure 2.1.4.



WebUI

WebUI Before Rules: ☑

WebUI Idle timeout: 900

WebUI HTTP port: 8080

WebUI HTTPS port: 443

Figure 2.1.4 Changing the HTTPS port

Demo mode and cOS Core licenses

When a Clavister Next Generation Firewall starts up, it will not have a license (unless previously installed). Without a license cOS Core will run in demo mode for 2 hours. When in demo mode the firewall can be used for trial and evaluation but some features will have limited functionality. Even though most of the recipes and solutions in this book could be configured without a license, we will assume that a valid license has been installed.

Once the 2 hour demo mode time limit has expired the firewall will become locked and it will need to be restarted in order to get a further 2 hours of usage.

The purpose of a license is to define what capabilities and limitations cOS Core will have. Such capabilities include such parameters like the number of VPN tunnels allowed and the maximum number of routing tables.

There are several ways to install a license:

1. Automatic activation from the WebUI.
2. Manual activation and upload using the WebUI.
3. Manual activation and upload using the CLI.
4. Manual activation and upload using InControl.

In this exercise we will be using method 2 because automatic activation is not possible in a virtual environment such as VMware™ and the aim is to be platform neutral in this book.

License registration and upload

To register a license we need to access www.clavister.com and either log in with an existing account or, if a new customer, create a new account.

After we have logged in we go to our account's license section and input the required data such as license key or service tag depending on what kind of license we have. Older Clavister firewalls use a license key plus MAC address to register, while newer firewalls use a service tag plus hardware serial number combination to register.

Once all the steps for registering a license are complete, we have the option to download the license file. The license file will be in the format of "<license-key>.lic".

There are two WebUI locations where we can upload a license file to cOS Core:

- **Status > Maintenance > License**
- **Status > Maintenance > Upgrade**

Browse to the location of our downloaded license file and click upload license. Once the license has been uploaded there will be two options:

- Finish and restart cOS Core (recommended).
- Finish and perform a *reconfigure* operation.

A reconfigure is similar to a restart when deploying a changed configuration. It is a soft restart which is much faster than a full restart of cOS Core. The reason why a restart is recommended is because memory for some parameters (for example, VPN tunnels) is only allocated after a restart, so to guarantee proper functionality a restart is recommended.

Recipe 2.2. Downloading configuration and system backups

Objectives

Configuration backups are very important for any system administrator. Having the ability to make a complete image of the configuration can be very useful and it can even be used to clone a configuration from one Clavister firewall to another. This recipe looks at these operations.

Detailed Discussion

It is highly recommended to perform a backup of the configuration and system at regular intervals. The backup feature in cOS Core can be found under **Status > Maintenance > Backup & Restore** in the WebUI.

There are two types of backups that can be performed:

- A configuration backup.
- A system backup.

A *configuration backup* consists of only the cOS Core configuration; cOS Core itself is not included. A *system backup* consists of the configuration, the cOS Core loader and the cOS Core executable firmware. It is very useful to have access to a full system backup before we attempt any firmware updates because we can then easily and quickly revert to the previous cOS Core state if it is required.

In case we want to clone a cOS configuration or restore a backup on a new Clavister firewall after a hardware replacement, it is important to remember that the license file will not be included in any backup file.

Performing a cOS Core update

Keeping cOS Core up to date with the latest patch or release version is essential in order to secure our network, as Clavister continuously improves and updates cOS Core.

It is likely that the cOS Core version pre-installed on the new Clavister hardware might already need an update. Due to the sensitive nature of where cOS Core is installed in a network (usually a central point), there is no automatic update of cOS Core. All updates have to be performed manually by the administrator.

New firmware updates and versions are downloaded from the Clavister corporate website after registering and logging in. Once we log in to our user account, select the new cOS Core version that we want to install and download the appropriate file that matches the hardware that we have. Upgrade files will have the extension *.UPG*.

Firmware upgrades are performed by selecting **System > Maintenance > Upgrade** in the WebUI, as shown below.

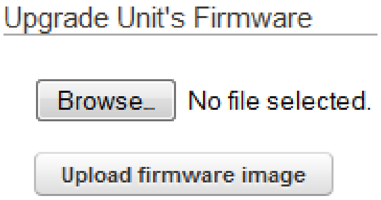


Figure 2.2.1 Uploading firmware

Important

Before performing any kind of firmware update it is highly recommended that a complete system backup is created. Extensive testing is always performed on any new cOS Core version to avoid problems but there may be new features and functions that might cause differences in the behavior of the firewall.

Make sure to consult the release notes for a new version before upgrading cOS Core. There may be a change that requires extra attention after the upgrade is complete.

Since a complete system backup contains both configuration and firmware files it is the fastest way to revert to the firewall's previous state.

Recipe 2.3. Internet access for administrators

Objectives

The objective of this recipe is to allow Internet access for the administrator from the management (Adm_net) network. This access is illustrated below in *Figure 2.3.1*.

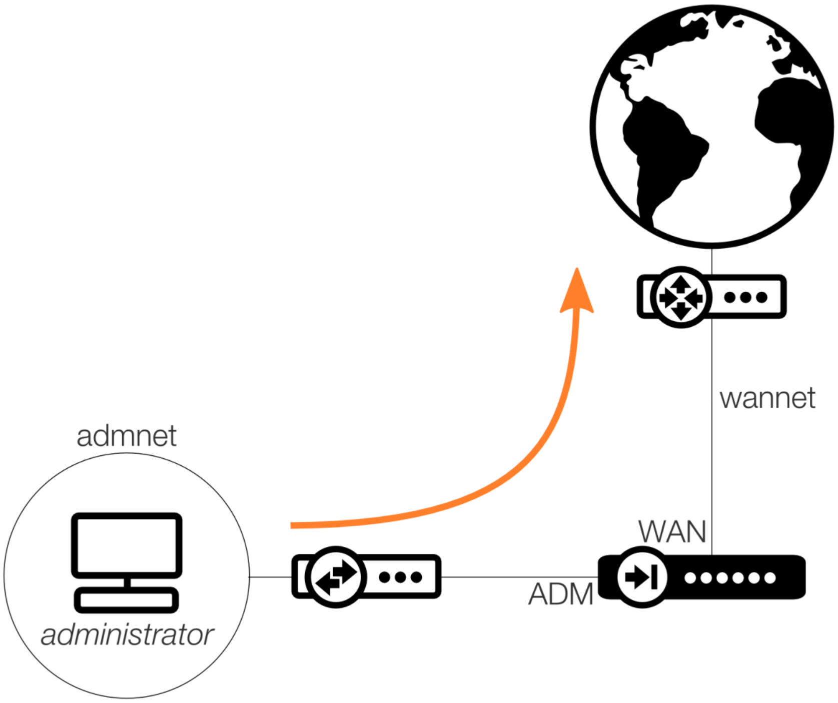


Figure 2.3.1 ADM to WAN network schematic with traffic direction arrow

Detailed Discussion

First, we need to decide which IP and network we want to assign for our various interfaces.

In our example, we will use the following networks for the various interfaces:

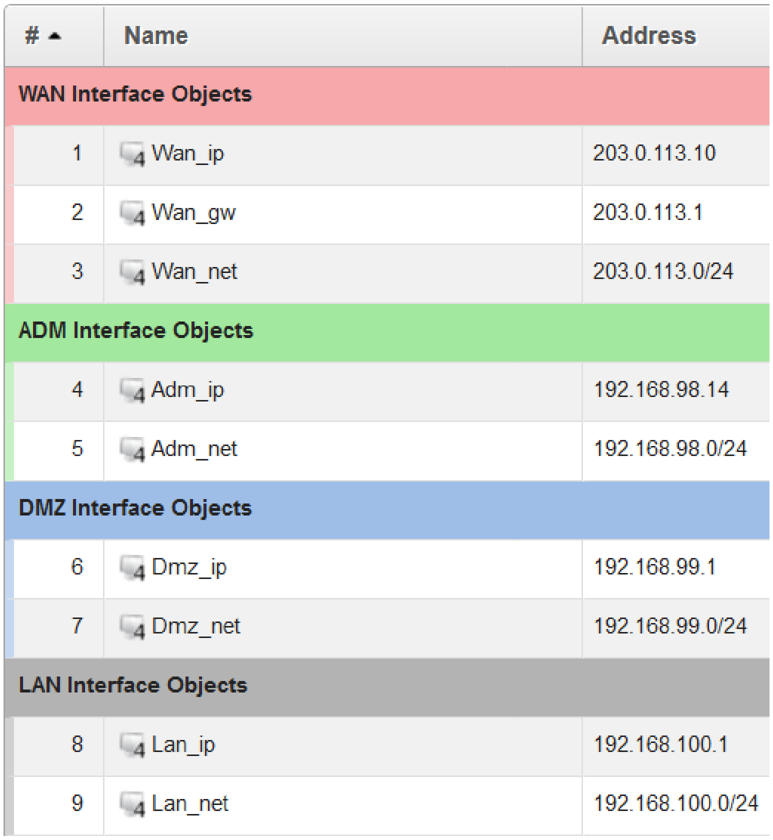
- **WAN** – 203.0.113.0/24
- **ADM** – 192.168.98.0.0/24

- **DMZ** – 192.168.99.0/24
- **LAN** – 192.168.100.0/24

A common practice is to select the IP address used by the interface itself and to use either the first or last usable IP in the network IP range (primarily because clients behind cOS Core will use it as their default gateway).

The administrator will not have much control over IP addresses received from their ISP and there may also be situations in our own internal network that will cause us to use an IP address different from the first or last one.

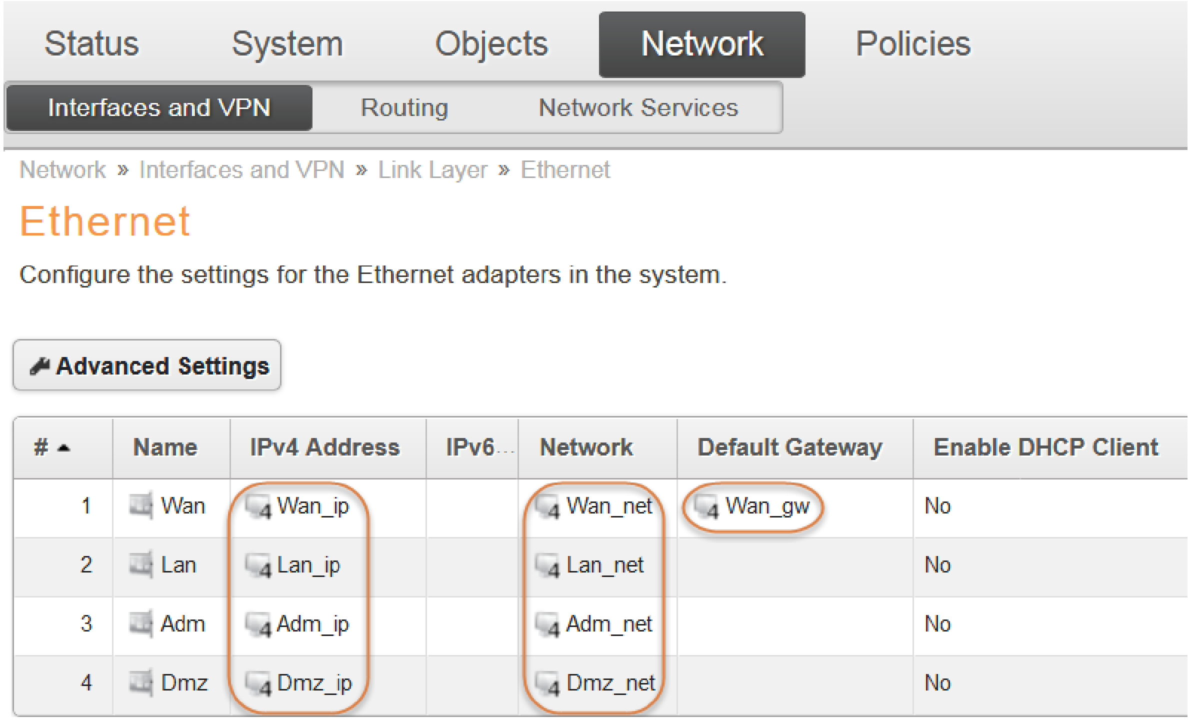
In the screenshot below, we have created IP and network objects that we will use on the various interfaces. These objects will be used and referred to in descriptions and in network schematics as we progress through this chapter.



# ▲	Name	Address
WAN Interface Objects		
1	Wan_ip	203.0.113.10
2	Wan_gw	203.0.113.1
3	Wan_net	203.0.113.0/24
ADM Interface Objects		
4	Adm_ip	192.168.98.14
5	Adm_net	192.168.98.0/24
DMZ Interface Objects		
6	Dmz_ip	192.168.99.1
7	Dmz_net	192.168.99.0/24
LAN Interface Objects		
8	Lan_ip	192.168.100.1
9	Lan_net	192.168.100.0/24

Figure 2.3.2 Address book object summary

Once all the required objects and networks are created, we now go to the interface section of the WebUI and make sure that all objects are assigned to the correct interface as shown in the screenshot below.



Ethernet

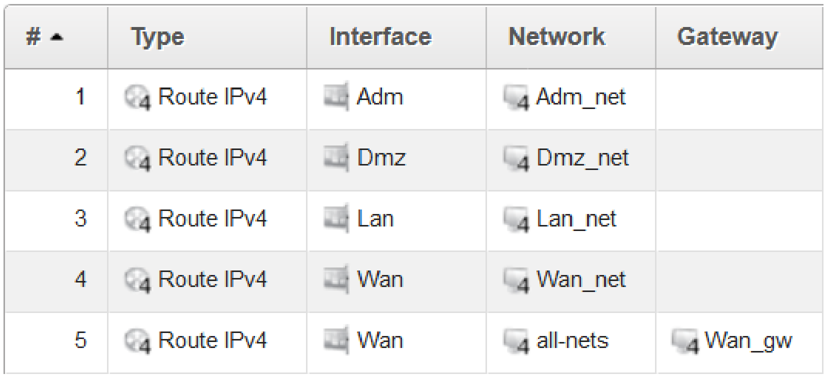
Configure the settings for the Ethernet adapters in the system.

# ▲	Name	IPv4 Address	IPv6...	Network	Default Gateway	Enable DHCP Client
1	Wan	Wan_ip		Wan_net	Wan_gw	No
2	Lan	Lan_ip		Lan_net		No
3	Adm	Adm_ip		Adm_net		No
4	Dmz	Dmz_ip		Dmz_net		No

Figure 2.3.3 Interface object allocation

We should note in the above that there is only one object which is set as the Default Gateway. This is the gateway address of our Internet Service Provider (ISP).

Once every object is added to the relevant interface(s), we will do a quick check of the routing table. Based on our current configuration, the routing table **main** should look like in Figure 2.3.4.



# ▲	Type	Interface	Network	Gateway
1	Route IPv4	Adm	Adm_net	
2	Route IPv4	Dmz	Dmz_net	
3	Route IPv4	Lan	Lan_net	
4	Route IPv4	Wan	Wan_net	
5	Route IPv4	Wan	all-nets	Wan_gw

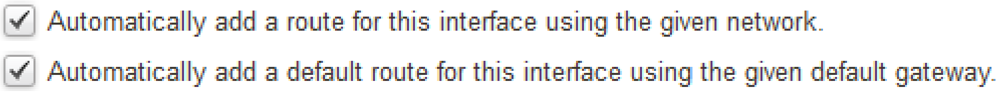
Figure 2.3.4 Routing table summary

Note

This routing table will look different depending on the amount of interfaces and the hardware platform.

Two Methods of Adding Routes

There are two ways to add routes to the routing table. One way is to enable the automatic route creation on each interface, as shown in the screenshot below. These checkboxes are enabled by default.



☑ Automatically add a route for this interface using the given network.

☑ Automatically add a default route for this interface using the given default gateway.

Figure 2.3.5 Automatic route creation options

This means that when we add a network object or gateway to an object like an Ethernet interface, a route will be created and added automatically.

The other method is to manually create a route in the routing table. One of the drawbacks of using the automatically created routes is that we cannot create comment groups on these routes. So it can be a good idea to uncheck the above checkboxes and create routes manually in order to be able to use comment groups. We will leave the options enabled because we will not do any more modifications to the routing table in this chapter.

IP Policies

Now we need to create the most important aspect of the recipe, the IP policies.

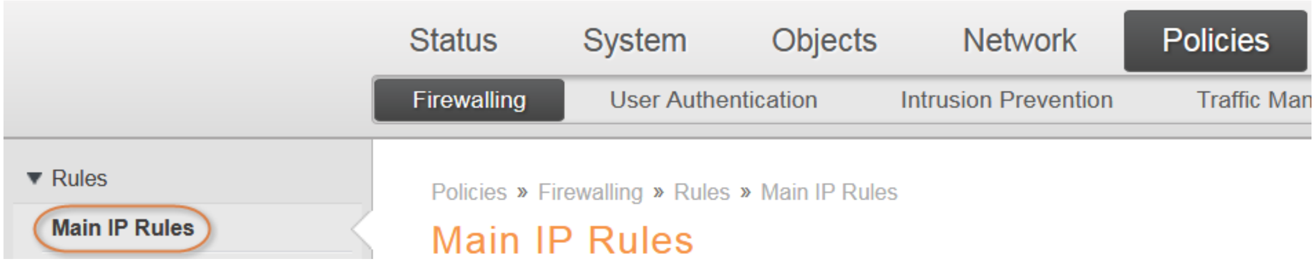


Figure 2.3.6 Main IP rules location

Normally, there would be a few IP policies auto-created (how many and how they are configured varies depending on which cOS Core version we are using). In this book, we will assume an empty IP rule set and create them from scratch.

In order to keep things simple we will create a very generous IP policy that will allow all users behind the ADM (administrators) interface to connect to any IP on the external WAN interface using any port and any protocol.

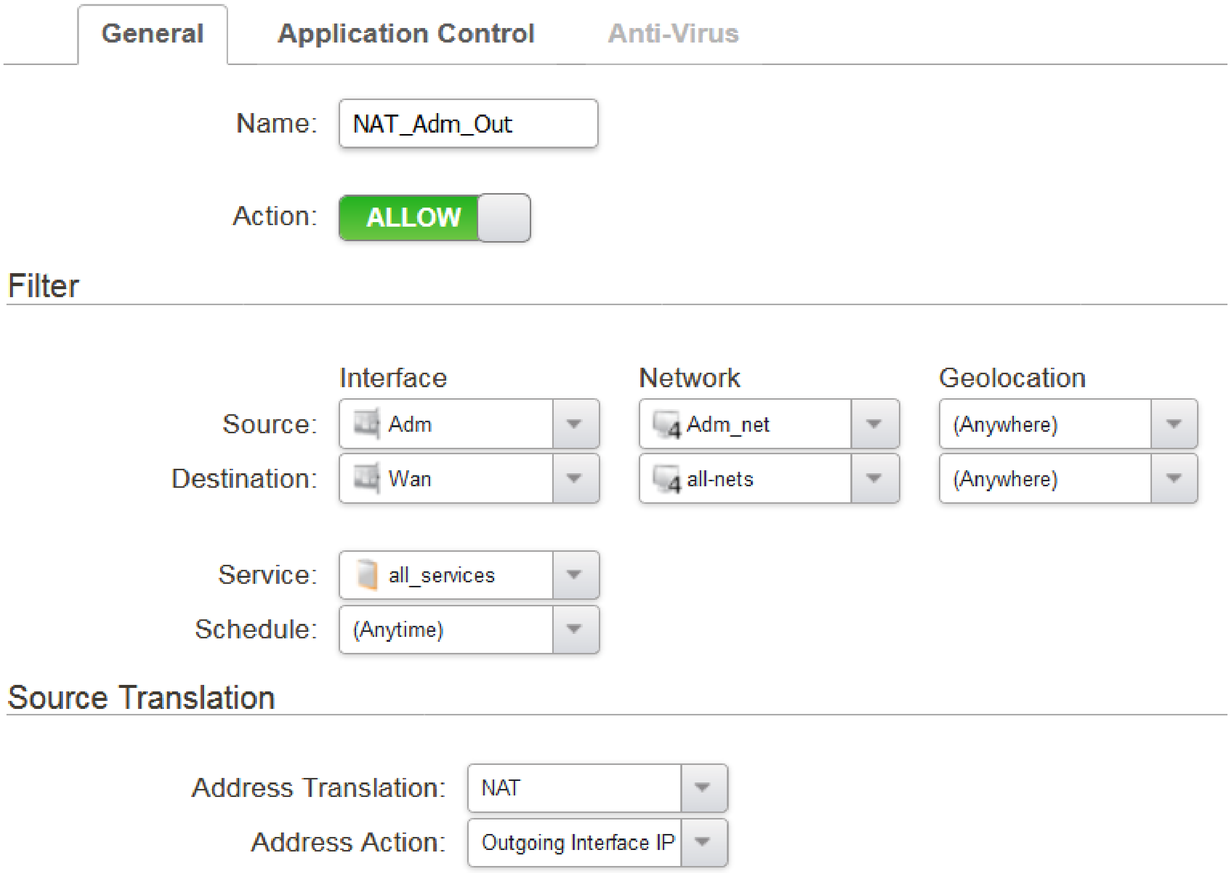
It would be reasonable to give the administrators temporary full external access in order for them to go to the Internet to download software updates and other programs needed in preparation for building and configuring the internal network. We say "temporary" because this access will become more restricted as we progress.

Here we will use a new policy feature called Source Translation and NAT (Network Address Translation).

Address Translation NAT means we allow the traffic from the defined source interface + source network to the destination interface + destination network but we mask the source IP of the traffic. This is normally used when clients on private IP addresses want to reach the Internet, their source IP addresses must be changed before leaving cOS Core.

The most common Address Action used when NATing is Outgoing Interface IP which means that the new IP we use to mask the traffic will be the Interface IP of the interface through which the traffic exits, in below example this is the Wan interfaces IP. In some scenarios you might want to use Single IP or NAT Pool as Address Action to manually choose which IP addresses to be used as the new source IP but in this book we will only be using Outgoing Interface IP.

The IP policy we will now create will have the properties shown in Figure 2.3.7.



Name: NAT_Adm_Out

Action: ALLOW

Filter

	Interface	Network	Geolocation
Source:	Adm	Adm_net	(Anywhere)
Destination:	Wan	all-nets	(Anywhere)

Service: all_services

Schedule: (Anytime)

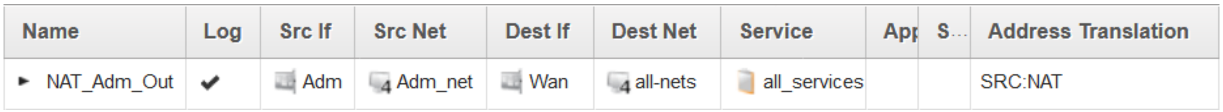
Source Translation

Address Translation: NAT

Address Action: Outgoing Interface IP

Figure 2.3.7 IP Policy properties

For future reference, we will primarily use the IP policy summary to describe IP policies and their properties unless special options are used. The policy summary for the above policy will look like the screenshot below.

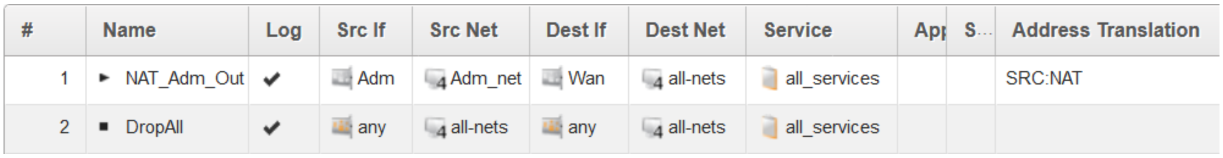


Name	Log	Src If	Src Net	Dest If	Dest Net	Service	App	S...	Address Translation
► NAT_Adm_Out	✓	Adm	Adm_net	Wan	all-nets	all_services			SRC:NAT

Figure 2.3.8 IP policy summary overview

The policy we have created will grant full access for everyone in any way connected to the ADM interface without any restrictions, all ports and protocols are allowed.

There is one more policy we would like to create before the end of this particular recipe. A policy called "DropAll" with the properties shown in the screenshot below.



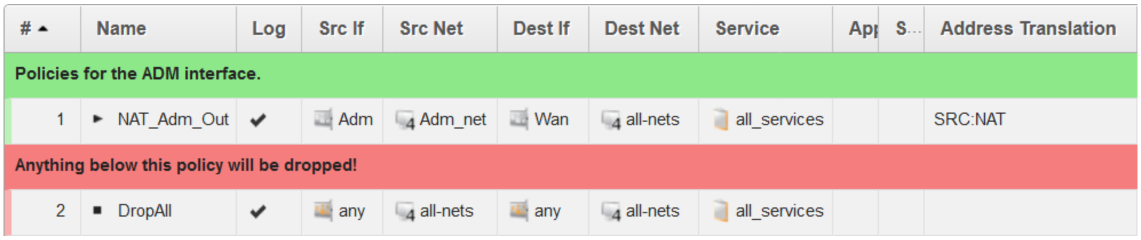
#	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Ap	S…	Address Translation
1	► NAT_Adm_Out	✓	Adm	Adm_net	Wan	all-nets	all_services			SRC:NAT
2	■ DropAll	✓	any	all-nets	any	all-nets	all_services			

Figure 2.3.9 The "DropAll" policy

Our DropAll does not have any restrictions. The action is set to Deny with the Deny Behavior Drop, source/destination interface is Any, source/destination network is all-nets and the service type is "all_services". This means everything on all interfaces and ports/protocols. Anything not being initiated from the ADM interface and network will be dropped by this policy.

Why use a DropAll policy?

Without the DropAll policy, traffic not matching our newly created policy will fall down to the bottom of the IP rule set. If there is still nothing there that matches, it will be dropped by a hidden default drop rule called "Default_Rule". This rule is identical to our DropAll policy but the advantage of having an explicit DropAll policy is that we can make it more distinct, give it a unique name, change its log category (if needed) and we can also use comment groups to make it clear that anything that reaches this policy will be dropped, as shown below.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Ap	S…	Address Translation
Policies for the ADM interface.										
1	► NAT_Adm_Out	✓	Adm	Adm_net	Wan	all-nets	all_services			SRC:NAT
Anything below this policy will be dropped!										
2	■ DropAll	✓	any	all-nets	any	all-nets	all_services			

Figure 2.3.10 A "DropAll" policy with comment groups

The DropAll policy is not a requirement, it is optional.

Again, it is highly recommended to use comment groups as soon as possible in order to start making our IP rule set easier to read and understand. So far, it is very easy to read but once we have hundreds or even thousands of policies and objects it will become harder.

Administrators can now access the Internet on all source and destination ports or protocols that they choose.

Recipe 2.4. Configuring the LAN interface for external and internal access

Objectives

The objective of this recipe is to give LAN users access to the Internet and also be able to reach servers behind the DMZ interface. By "LAN users" we mean a user located behind the interface called LAN. This person might also be referred to as the *client*, *student* or *employee*. The majority of the people using the network will be located behind the LAN interface, as shown in *Figure 2.4.1* below.

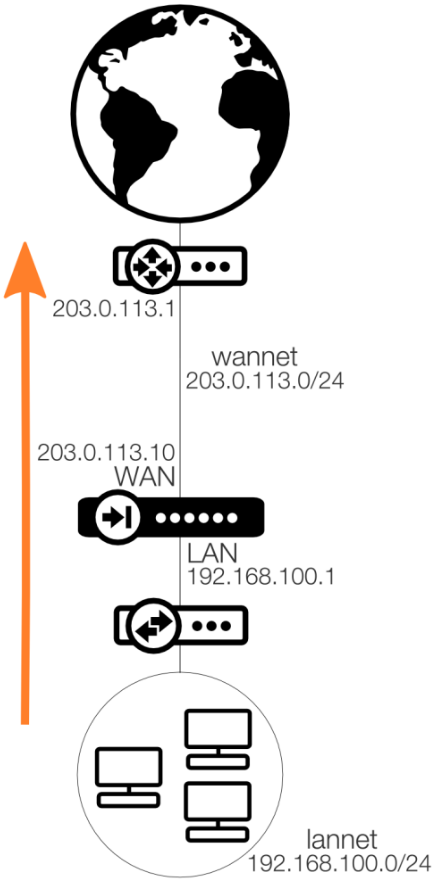
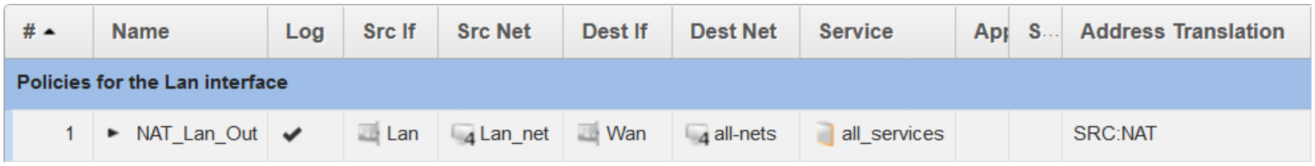


Figure 2.4.1 LAN to WAN network access

Detailed Discussion

Since all the basic networks and IPs have already been defined, we only need to create a few extra IP policies. No additional routes or changes to existing routes are needed.

We start by giving Internet access to our users behind the LAN interface by creating an IP policy like the one shown below.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	App	S...	Address Translation
Policies for the Lan interface										
1	► NAT_Lan_Out	✓	Lan	Lan_net	Wan	all-nets	all_services			SRC:NAT

Figure 2.4.2 A NAT IP policy for the LAN interface

And this is all that is needed. This IP policy is using the action NAT as we are going from a private network to something on the Internet. By creating this IP policy, we give our users behind the LAN interface access to the Internet using the external WAN interface on any port and any protocol. We set the destination interface to be WAN as we want to limit the number of destination interfaces on which this IP policy could be triggered. This policy will now only trigger if the destination interface is the external WAN interface (in other words, for Internet traffic).

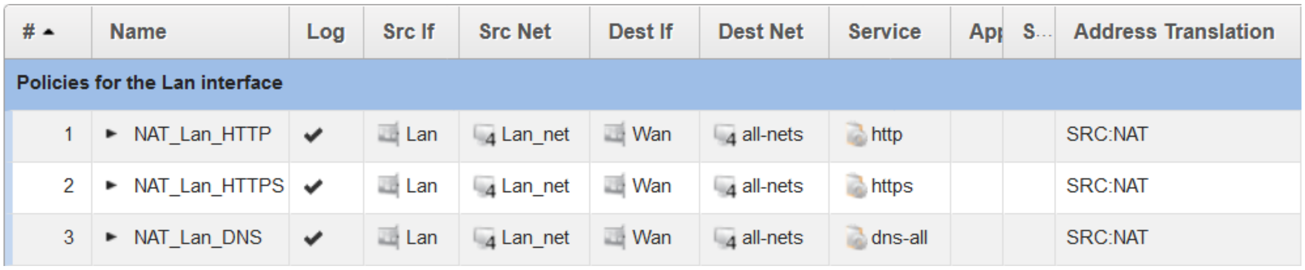
Note

Don't forget to place any newly created IP policy above the DropAll policy in the rule set because IP policy ordering is very important.

However, even though this works it is not recommended to configure the IP policy like this, but many administrators choose to do so anyway. We will explain why this is not a good idea. The main reason is security. By using the service "all_services", we do not restrict the external access at all. Users are free to use whatever application and program they want to access the Internet. Exploits, malware, keyloggers, bots etc. Anything a client's machine may get infected with will have full access to do whatever they want.

Even if many of these types of programs may use default ports such as 80 or 443, there is no reason to open up full access towards the Internet. It is easy to configure, yes, but not secure or recommended.

It is recommended to make an evaluation of what exactly can be initiated from the LAN interface towards the Internet. If we look at the absolute basics, it is reasonable to allow the following services: HTTP, HTTPS and DNS, as shown in the screenshot below.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Ap	S...	Address Translation
Policies for the Lan interface										
1	► NAT_Lan_HTTP	✓	Lan	Lan_net	Wan	all-nets	http			SRC:NAT
2	► NAT_Lan_HTTPS	✓	Lan	Lan_net	Wan	all-nets	https			SRC:NAT
3	► NAT_Lan_DNS	✓	Lan	Lan_net	Wan	all-nets	dns-all			SRC:NAT

Figure 2.4.3 NAT IP policies for specific services

It is possible to create a service group in the address book and add "http, https and dns-all" to that group. Then we only need one IP policy for all 3 services. It is a matter of personal preference which method is used. In this book, we will create one IP policy per service for simplicity.

Accessing the DMZ from the LAN interface

We now have policies for basic Internet access. But since we have restricted these policies to only trigger if the target interface is the external (Wan), they will not match if we try to reach something in, for example, the DMZ. This is illustrated below in *Figure 2.4.4*.

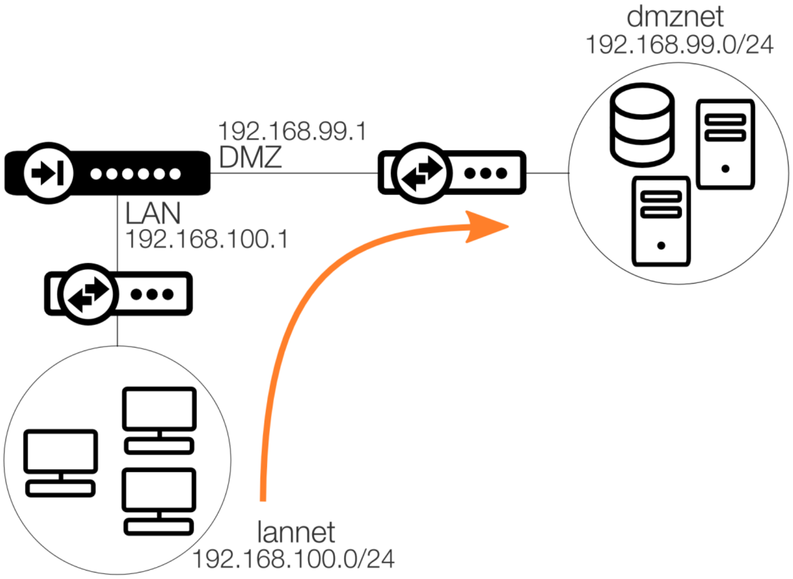
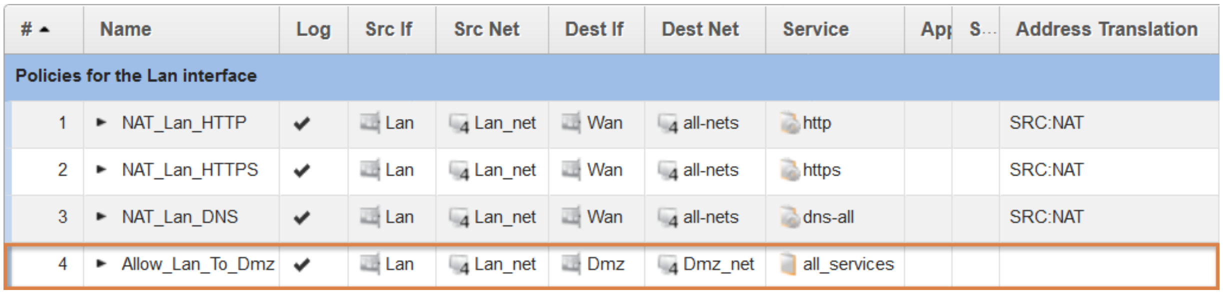


Figure 2.4.4 Traffic from LAN to DMZ with traffic direction arrow

We could modify our existing policies to trigger for access on the DMZ interface as well, but it is better to create new policies for access between multiple interfaces. Some reasons why this is a good idea are the following:

- Better overview of policy functionality.
- Less chance of accidental access to restricted resources.
- Easier to customize functionality.

In order to allow users to initiate connections from the LAN to the DMZ network, we need an IP policy that looks like policy **4** in the screenshot below.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	App	S...	Address Translation
Policies for the Lan interface										
1	► NAT_Lan_HTTP	✓	Lan	Lan_net	Wan	all-nets	http			SRC:NAT
2	► NAT_Lan_HTTPS	✓	Lan	Lan_net	Wan	all-nets	https			SRC:NAT
3	► NAT_Lan_DNS	✓	Lan	Lan_net	Wan	all-nets	dns-all			SRC:NAT
4	► Allow_Lan_To_Dmz	✓	Lan	Lan_net	Dmz	Dmz_net	all_services			

Figure 2.4.5 LAN to DMZ IP policy

Note that policy **4** in the above rule set is not an address translation policy (both source and destination translation are set to None). The reason for this is because there is no need to mask the source sender IP for communication between two internal private networks. It is however up to the administrator to decide what kind of actions and services he wants to use on the policies and what kind of requirements there are in the overall network design.

Why use *all_services* on the policy towards the DMZ?

This is a matter of preference and it is up to the administrator to decide how much access should be allowed between the internal networks.

Many would prefer to allow full communication between their own internal networks. Some administrators even use Stateless policies for this communication. It is however recommended to use state tracked policies whenever possible (Stateless policies are not state tracked).

Should users behind the LAN interface have access to the DMZ at all?

There is a higher chance that a machine in the LAN gets infected by malicious programs than a server in the DMZ. Restricting access to the DMZ as much as possible is recommended.

Is there a reason for a normal user to access the servers in DMZ directly?

It all depends on how the network is designed and what kind of applications and content are being run in the DMZ. Instead of allowing the entire LAN network access to the DMZ with all ports and protocols, it would be better to provide limited access to the DMZ for specific computers. One way to do that is to use network object groups.

Further restricting the access to only allow specific services such as HTTP, DNS etc. is even better. Any access between two interfaces that allows all ports/protocols or the entire network is a potential security risk.

Creating and using network object groups for customized access to the DMZ

In *Figure 2.4.6* below, we have 3 client PCs with static IP addresses that should be given access to the DMZ from the LAN interface. We want to move away from our previous policy of allowing the entire network behind the LAN interface to reach the DMZ.

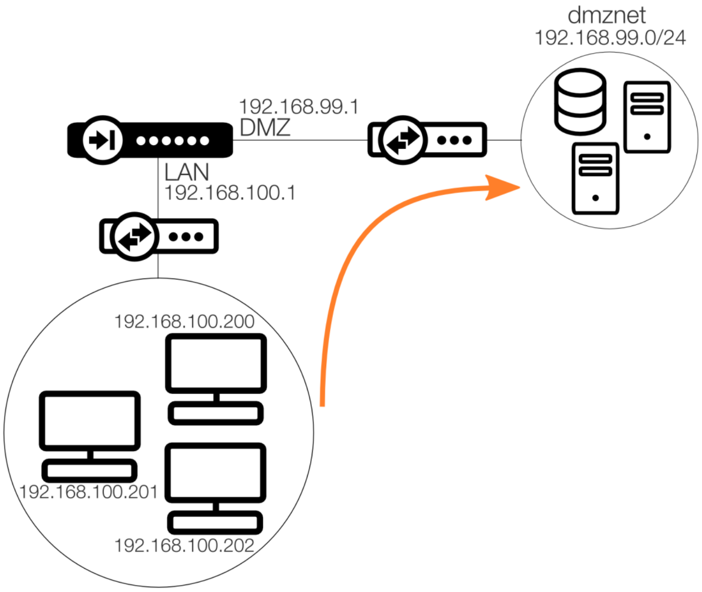
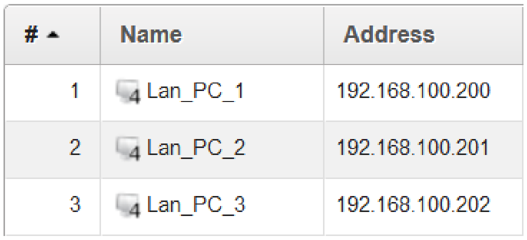


Figure 2.4.6 Access from hosts in LAN to the DMZ network

The above shows three specific computers behind the LAN interface that will be given access to the servers behind the DMZ interface. The orange arrow shows the direction we want to be able to initiate the traffic.

To accomplish this, we create 3 new address objects that contain the IP addresses of the machines behind the LAN interface, as seen in figure 2.4.7. These machines will be given access to the DMZ and they will have static IP address configured.



#	Name	Address
1	Lan_PC_1	192.168.100.200
2	Lan_PC_2	192.168.100.201
3	Lan_PC_3	192.168.100.202

Figure 2.4.7 LAN host objects with corresponding addresses

Next we create an address group object to contain our newly created network objects as shown below.

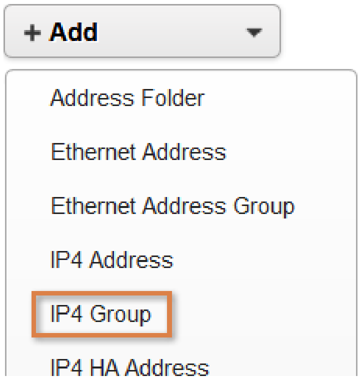
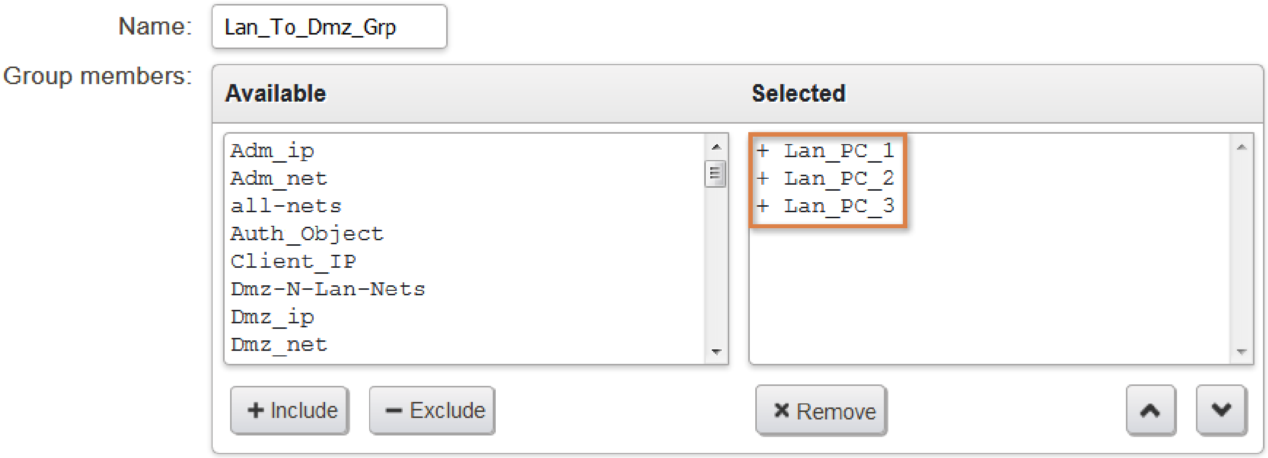


Figure 2.4.8 Adding a network group object to contain our 3 machines in LAN

Then we give the group an appropriate name and include our objects in the group, as shown below.



Name: Lan_To_Dmz_Grp

Group members:

Available

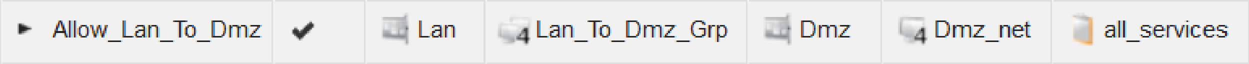
```
Adm_ip
Adm_net
all-nets
Auth_Object
Client_IP
Dmz-N-Lan-Nets
Dmz_ip
Dmz_net
```

Selected

```
+ Lan_PC_1
+ Lan_PC_2
+ Lan_PC_3
```

Figure 2.4.9 Adding network objects to a group object

Once the group object is complete, we change our Lan_to_Dmz policy to use our newly created group object as source network instead of the entire Lan_net (192.168.100.0/24) object. This is shown in the screenshot below.



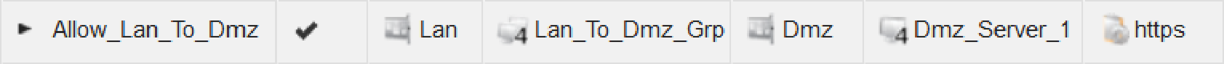
► Allow_Lan_To_Dmz	✓	Lan	Lan_To_Dmz_Grp	Dmz	Dmz_net	all_services

Figure 2.4.10 Using a group with an IP policy

The effect of this change is that only the three PCs behind the LAN interface will have access to the DMZ network. If anyone else tries to reach the DMZ from the rest of the LAN network, the connection attempt will be dropped.

Currently we allow all ports and protocols from these machines to the entire DMZ network, but this can be changed as well. We can limit access to only allow specific ports and also to only allow access to one or more specific IP addresses in the destination network in a similar way to what we did with the connecting client PCs.

As an example, let us configure a really secure policy for access from LAN to DMZ with the policy below.



► Allow_Lan_To_Dmz	✓	Lan	Lan_To_Dmz_Grp	Dmz	Dmz_Server_1	https

Figure 2.4.11 Example of a secure policy when giving access to the DMZ

Now, we have created a new address book object called "Dmz_Server_1" which is a single IP address to a server in the DMZ. We use that object as the destination network for the IP policy and also limit the ports that can be used to only one: HTTPS port 443. This is really secure and the chance of unauthorized access to this server is very low.

These are just some of the basic things we can do in the Clavister firewall. We could also add layers upon layers of verifications and authentication to even further restrict access. We could require a username and password, restrict access to only certain times or days of the week, require an encrypted Virtual Private Network (VPN) connection, specify which client MAC addresses are allowed, limit which applications that are allowed to run on the designated port and much more.

In the end it will be up to the administrator to decide how he would like to restrict communication between the various networks and interfaces, but cOS Core provides the tools needed.

Tip

When creating new policies there is a very handy feature called "Clone". When we setup a new policy, (or service, address book object etc.) there is a fairly good probability that it will be based on an existing policy with some minor changes in parameters.

In those situations we can right-click an existing policy and select "Clone" to clone that particular policy then give it a new name and parameters. This is shown in the screenshot below.

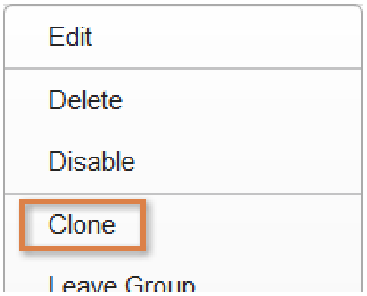


Figure 2.4.12 Policy/object cloning

Note that an object clone is always added ***at the end*** *of the current list of objects.*

Recipe 2.5. Configuring a DMZ and allowing access to an internal web server

Objectives

The objective of this recipe is to allow Internet access to the server(s) and other computers located in the DMZ.

It would be reasonable to provide computers in the DMZ with at least some external access in order to download such things as system updates or new Anti-Virus signatures.

Figure 2.5.1 presents an overview of networks we will use in this recipe and the orange arrow indicates the traffic direction path we are interested in.

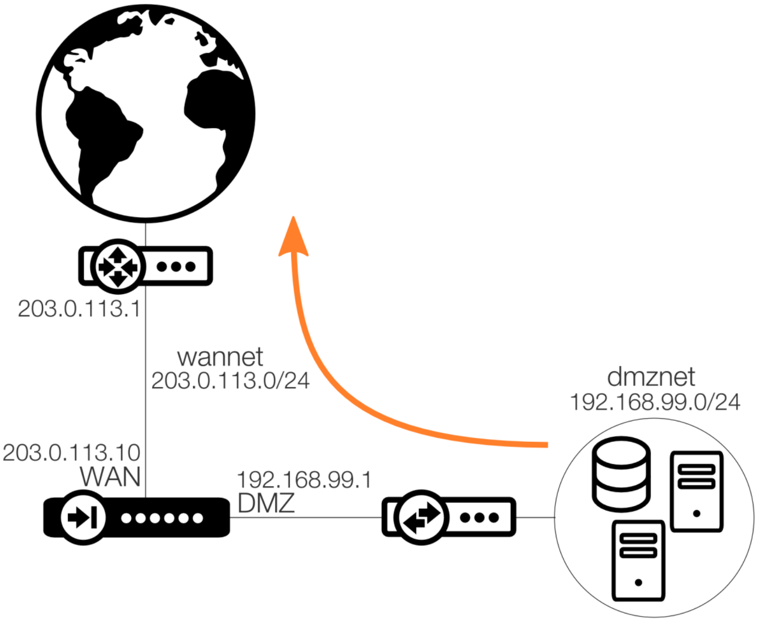


Figure 2.5.1 DMZ to WAN network with traffic direction arrow

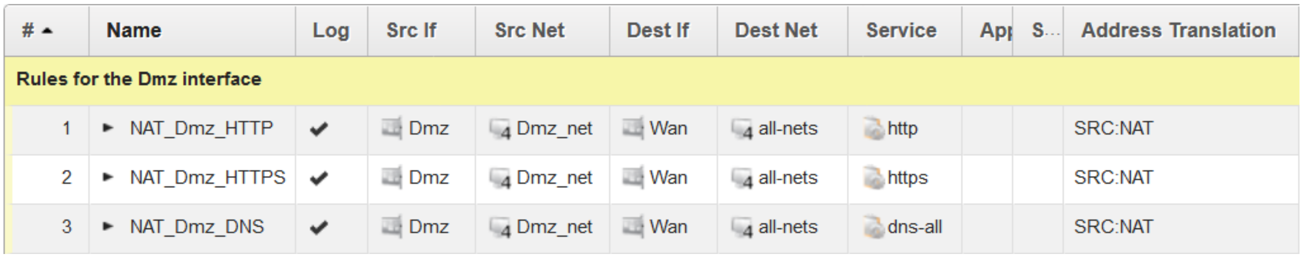
We will also create policies needed for users on the Internet to be able to reach a web server located behind the DMZ interface.

Detailed Discussion

All the basic networks and IPs have been already defined in earlier recipes. We only need to create a few extra IP policies. No additional routes or changes to existing routes are needed.

The main difference here is that the policies will now allow traffic from the DMZ interface instead of LAN.

The screenshot below shows the policies we have created to allow the servers in the DMZ to connect to the Internet using the HTTP, HTTPS and DNS protocols.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Ap	S…	Address Translation
Rules for the Dmz interface										
1	► NAT_Dmz_HTTP	✔	Dmz	Dmz_net	Wan	all-nets	http			SRC:NAT
2	► NAT_Dmz_HTTPS	✔	Dmz	Dmz_net	Wan	all-nets	https			SRC:NAT
3	► NAT_Dmz_DNS	✔	Dmz	Dmz_net	Wan	all-nets	dns-all			SRC:NAT

Figure 2.5.2 DMZ to WAN IP policies that allows servers on DMZ to access Internet using HTTP, HTTPS and DNS

Configuring policies for external access to an internal server in the DMZ

Our objective is to allow clients on the Internet access to a Web Server located in the DMZ.

So far we have only created policies allowing connections towards the DMZ interface when the connections are being initiated from internal networks located on the LAN interface. Now, we will create policies that enables us to reach a web server located on the DMZ interface from the Internet.

The diagram below illustrates what we are trying to achieve, with the orange arrow indicating the traffic direction we want to use.

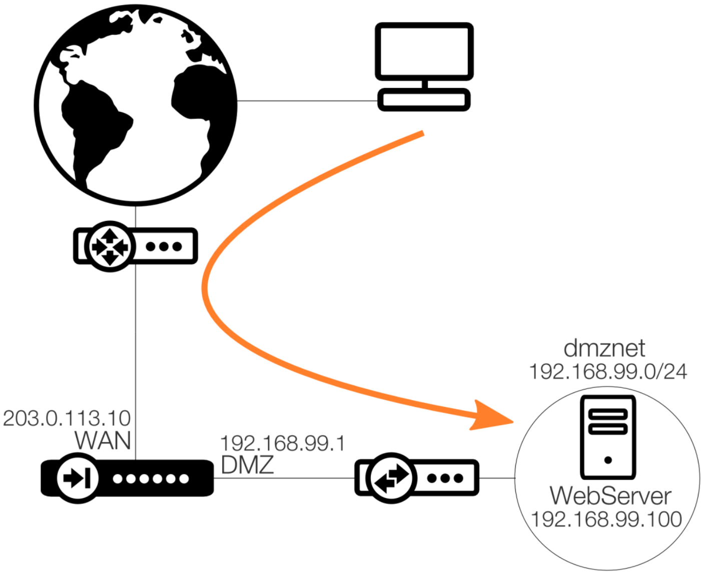
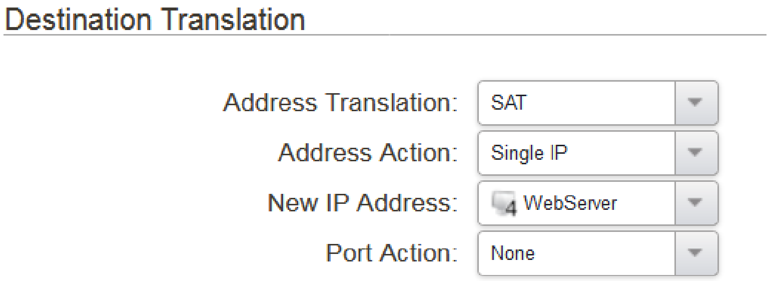


Figure 2.5.3 Connecting to internal web server on DMZ from a client on the Internet

To accomplish this we will use a policy feature called Destination Translation and SAT (Static address translation). The policies we have used so far have been using Source Translation (NAT (Network Address Translation)) or no address translation at all.

Destination Translation can be found just below Source Translation in the IP policy configuration. We will only be using Address Action Single IP in the examples of this book. In the screenshot below is an example of what Destination Translation and SAT can look like.



Destination Translation

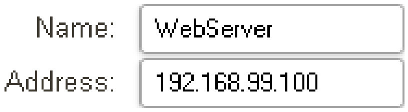
Address Translation:	SAT
Address Action:	Single IP
New IP Address:	WebServer
Port Action:	None

Figure 2.5.4 Configuring Destination Translation and SAT

Note

Some network equipment vendors use the term "port forwarding" when referring to SAT. Both terms refer to the same functionality.

In order to keep things structured we create a new IP address object in the address book we call "Web Server" with the properties shown in the screenshot below.

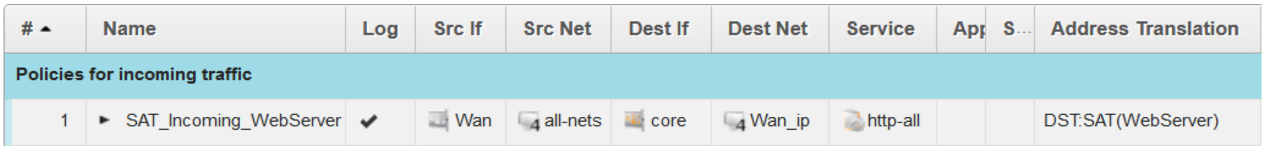


Name: WebServer

Address: 192.168.99.100

Figure 2.5.5 The IP address of the web server

Using the server object above as the New IP Address for the destination translation, the next step is to create a SAT policy like the one shown in the screenshot below.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	App	S...	Address Translation
Policies for incoming traffic										
1	► SAT_Incoming_WebServer	✔	Wan	all-nets	core	Wan_ip	http-all			DST:SAT(WebServer)

Figure 2.5.6 SAT policy definition

Using Core as the Destination Interface

One consideration that is important to highlight in the above definition is the destination interface. Here, we have selected Core as the target interface but what exactly is the Core interface?

Specifying *Core* as the destination interface indicates that it is cOS Core itself that must respond to traffic arriving on this interface. Examples of the use of the *Core* interface are when cOS Core responds to an ICMP "Ping" request or when it acts as a PPTP or L2TP server.

By specifying the destination interface of a route as *Core*, cOS Core will know that it is itself that is the ultimate destination of the traffic. There are some exceptions to this rule but we will not go into details regarding those in order to avoid confusing things at this point.

In our current setup, all IP addresses that are configured on the various physical interfaces are automatically *Core routed*. This means IP addresses 192.168.98.14, 192.168.99.1, 192.168.100.1 and 203.0.113.10, and lastly the localhost IP 127.0.0.1 all have routes which have the route's interface set to *Core*.

The IP address object we created for the web server (192.168.99.100) is, therefore, not a Core routed IP.

Using the WAN IP as destination network

As shown above, we are using a source interface of WAN and a destination network of Wan_ip. This can be a bit confusing as both the source and destination are seemingly the same. We have to keep in mind the traffic direction and how the policy is interpreted by cOS Core.

From a client perspective, the client wants to simply connect to our web server at a specific IP address. In this case, it is the address Wan_IP. From cOS Core’s perspective, we need to create a policy that allows the traffic the client wants to initiate towards our server on the inside.

Let's list out what is happening:

- **Src if:** When traffic is sent from the client it will arrive on the Internet interface, Wan. The source interface on the policy must then be WAN (or Any, but that is much less secure) as this is the interface that traffic is arriving on.

- **Src net:** We do not know the source IP of the client. There can be thousands, if not millions, of different source IPs that want to connect to our web server. We have chosen "all-nets" as the source network due to this reason which means all IPv4 IP addresses.

- **Dest if:** The destination interface will be the interface where "Wan_ip" is routed since this IP belongs to the Core interface and, as previously explained, the destination interface has to be Core (it could be Any, but that is insecure).

- **Dest net:** This is the IP address to which the client wants to connect and is basically the end of the journey for the client's request. In our example, it is the "web servers" public IP address, which is Wan_ip.

Now, clients on the Internet will be able to access the web server located behind the DMZ interface by connecting to the external public IP address of the Clavister firewall (203.0.113.10). In the background, however, the destination address will be translated to connect to the internal web server (192.168.99.100) located behind the DMZ interface.

Recipe 2.6. Accessing the DMZ web server from the internal (LAN) network

Objectives

The objective of this recipe is to configure our IP policies so that users behind the LAN interface (and part of the Lannet) would be able to connect to the corporate webpage/server located on the DMZ by connecting to the external interface, as shown next in *Figure 2.6.1*.

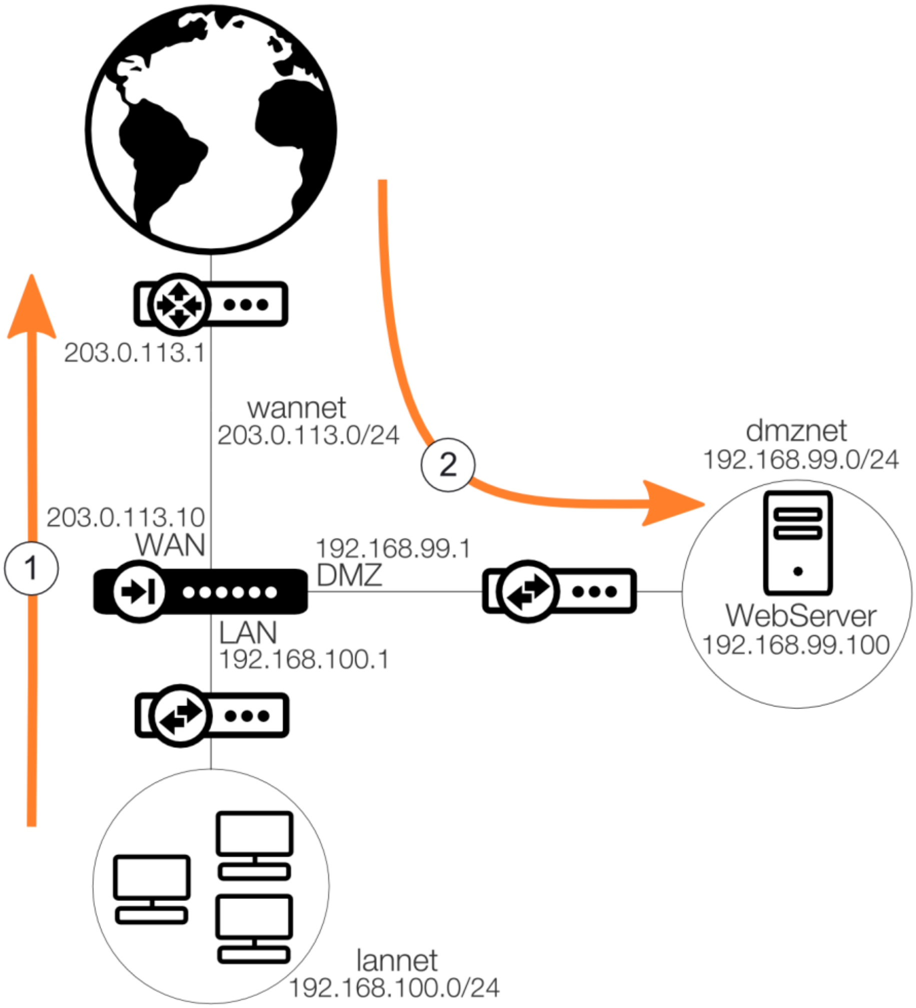


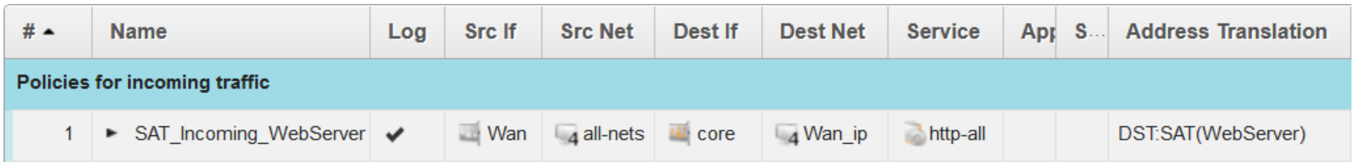
Figure 2.6.1 DMZ web server connection from LAN client via a public IP

Detailed Discussion

Previously, we configured access to the DMZ interface from the LAN interface using a normal allow policy. This works fine if the connecting client knows the private IP address of the web server and then connects directly to it.

However, this is a very unusual situation. If we have a corporate webpage, we do not connect to it by typing in its private IP address such as 192.168.99.100. Instead, we type in the name of the webpage name such as *www.clavister.com*.

Unless we have configured an internal DNS server in such a way that it replies with the private IP address of the webpage, we need to make some modifications to our incoming IP policy. After previous recipes, our incoming rule set now looks like the one shown below.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Ap	S...	Address Translation
Policies for incoming traffic										
1	► SAT_Incoming_WebServer	✔	Wan	all-nets	core	Wan_ip	http-all			DST:SAT(WebServer)

Figure 2.6.2 Incoming policy to DMZ from Internet

The problem is that the policy will not trigger when we initiate a connection from the source interface LAN. This policy will only trigger if traffic arrives on the WAN interface.

To solve this problem, we create an interface group consisting of LAN and WAN.

Interface groups can be found in the WebUI by going to **Network > Interfaces** and then selecting **VPN > Misc > Interface Groups**.

This is shown in the next screenshot from the WebUI.

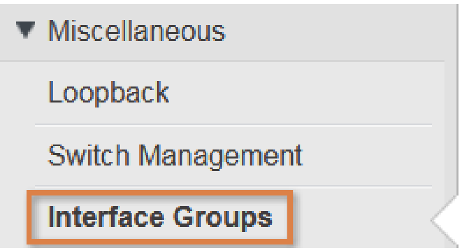
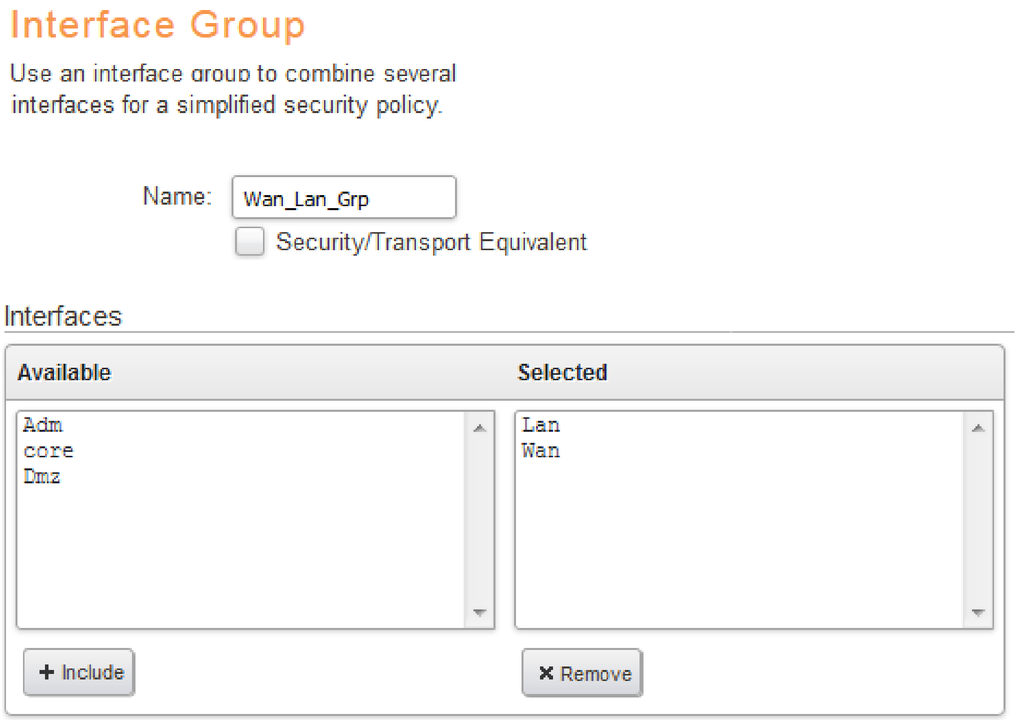


Figure 2.6.3 Interface groups

Next, add LAN and WAN to our interface group as shown below.



Interface Group

Use an interface group to combine several interfaces for a simplified security policy.

Name: Wan_Lan_Grp

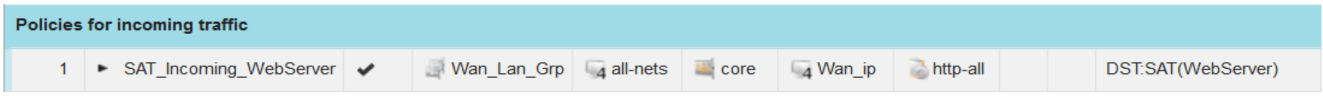
□ Security/Transport Equivalent

Interfaces

Available	Selected
Adm core Dmz	Lan Wan

Figure 2.6.4 Creating an interface group

Next, use this interface group as source interface in our IP policy, as shown in the screenshot below.



Policies for incoming traffic

1	►	SAT_Incoming_WebServer	✔	Wan_Lan_Grp	all-nets	core	Wan_ip	http-all		DST:SAT(WebServer)

Figure 2.6.5 Incoming policy modified to also trigger on the LAN interface

Now we will be able to connect to the company webpage no matter if we are located on the Internet or behind the LAN interface.

Note

Even though we do not include the ADM interface in this group in this example, it might be a good idea to do so to give administrators the same level of access.

This is not the only way to solve this particular scenario. We can also create a new SAT policy that specifically triggers for the LAN interface. Or we can modify the SAT policy and the outgoing policy for the LAN interface to include the Core interface. cOS Core is very flexible and there are, in most cases, multiple ways to achieve an objective.

Recipe 2.7. Accessing the web server from the same interface as the web server itself

Objectives

There exists a very common configuration scenario that can occur when connecting to the company webpage, if the client and server are located behind the same interface and network.

This recipe will discuss the problem and how to solve it. *Figure 2.7.1* below, illustrates the situation where there is a client located on the DMZ interface that wants to access the web server located in the DMZ by connecting to a Fully Qualified Domain Name (FQDN). An example of an FQDN is *myserver.example.com*.

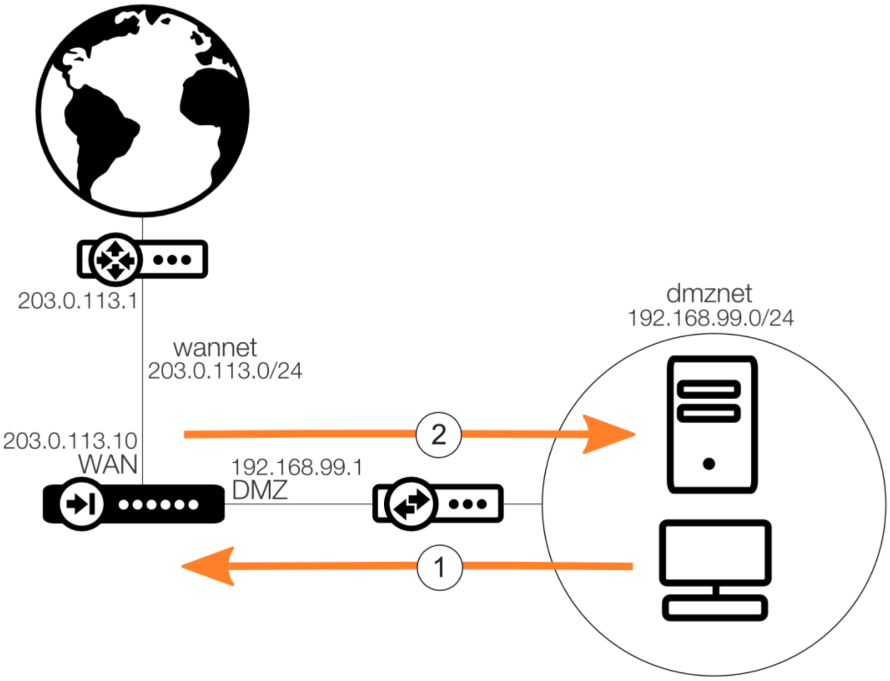
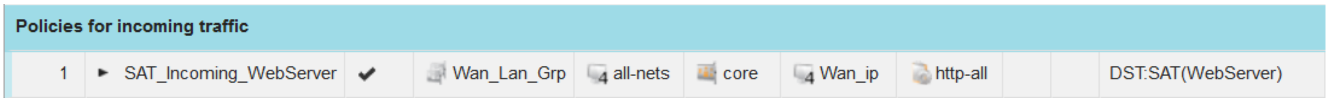


Figure 2.7.1 Initiate a connection from DMZ to server in DMZ

Detailed Discussion

The connecting clients want to reach the company webpage by typing in its web address (such as www.clavister.com). This in turn will result in a public IP that needs to be handled within the rule set described previously (*Recipe 2.7. Accessing the web server from the same interface as the web server itself*).

In the current configuration example, we do not have any policies that handle this particular situation. Our rule set currently looks like the following for the incoming policies to the server in DMZ:



Policies for incoming traffic

1	► SAT_Incoming_WebServer	✔	Wan_Lan_Grp	all-nets	core	Wan_ip	http-all			DST:SAT(WebServer)

Figure 2.7.2 The IP policy set for incoming traffic

Based on the source interface group we are using in the IP policy, we allow traffic coming from both LAN and WAN. As we are initiating traffic from the DMZ interface in this scenario it means that the policy will not trigger.

An easy solution would be to add the DMZ to our existing interface group object.

The problem

But if we add the DMZ interface to the interface group and deploy these changes, we would still be unable to reach the web server from the DMZ. Why is that?

The reason is due to a packet direction problem which is illustrated below.

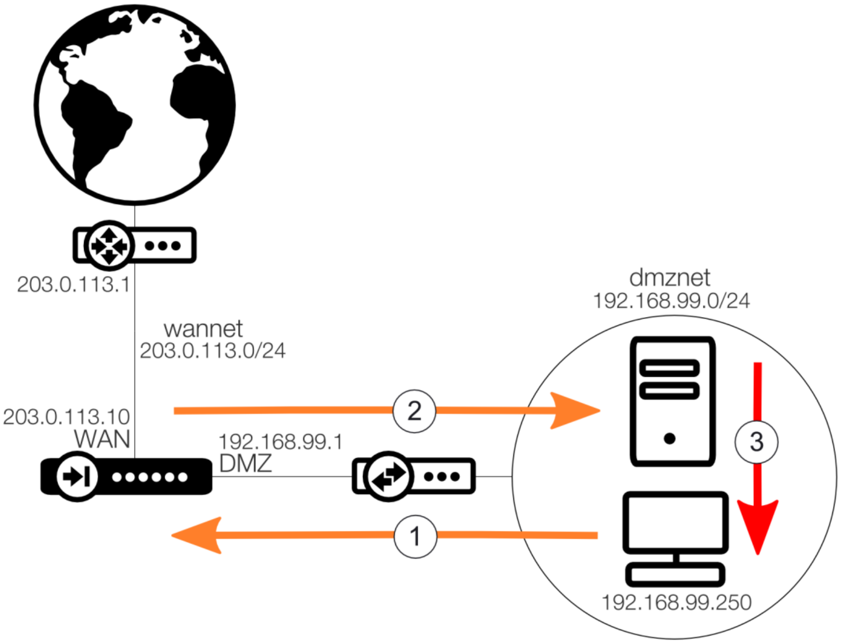


Figure 2.7.3 The packet direction problem

The event flow will be as follows:

- A client connects to the public IP address by typing in the web server's public FQDN name (**1**).
- Our IP policy triggers and forwards the client request to the server in DMZ (**2**).
- The server on DMZ gets a request from IP 192.168.99.250 which is part of the web server's own network range (192.168.99.0/24).
- Since the IP is part of the server's own IP range, it will simply send the reply directly to the connecting client (**3**).

- The connecting client gets a reply from a private IP address when it is waiting for a reply from a public address as resolved by the DNS name.

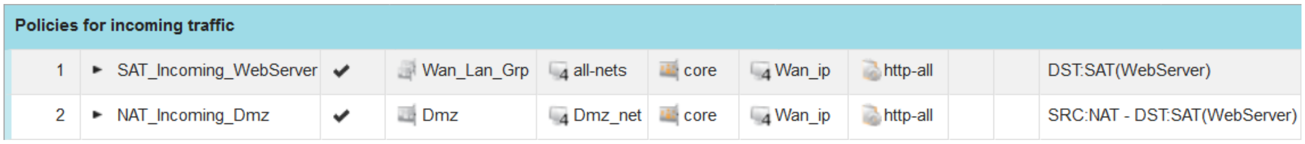
- The client gets confused by this as it receives an answer from an unexpected IP address and drops the packet. The connection attempt fails.

To further clarify this issue with another example: If a client sends a request to IP 203.0.113.50, it will expect an answer from this same IP. If it instead gets a reply from 192.168.99.100, it will simply discard this packet and keep waiting for a reply from 203.0.113.50.

The solution

The solution is to hide the client's true source IP from the server and making the web server believe it should reply to the firewall instead of the client directly.

This is accomplished by placing a second policy for the Dmz that uses both NAT and SAT instead of adding the DMZ interface to the interface group as shown below.



Policies for incoming traffic

1	►	SAT_Incoming_WebServer	✓	Wan_Lan_Grp	all-nets	core	Wan_ip	http-all		DST:SAT(WebServer)
2	►	NAT_Incoming_Dmz	✓	Dmz	Dmz_net	core	Wan_ip	http-all		SRC:NAT - DST:SAT(WebServer)

Figure 2.7.4 DMZ NAT policy solution

This means that when the DMZ client connects, policy #2 will trigger instead of #1.

If we come from a host on the Internet (let's say 203.0.113.5), since 203.0.113.5 is not part of our DMZ network and the traffic is not coming from the DMZ interface, it will not trigger policy number 2. Instead, policy number 1 will trigger as this is an IP address that exists (is routed) behind the external Internet interface and does not exist behind Dmz.

If we arrive from a host on the DMZ (let's say 192.168.99.250) and in this case the IP we are coming from 192.168.99.250 is part of the DMZ network so policy number 2 will trigger.

This NAT policy will mask the client's true IP address, so when the packet is arriving on the web server it will have the IP of the DMZ interface as sender (192.168.99.1). The web server will reply to this address and then the entire conversation between the connecting client and web server will pass through cOS Core, keeping the communication connection intact between the client and the server. The client gets a reply from the IP it sent the request to, as shown in Figure 2.7.5.

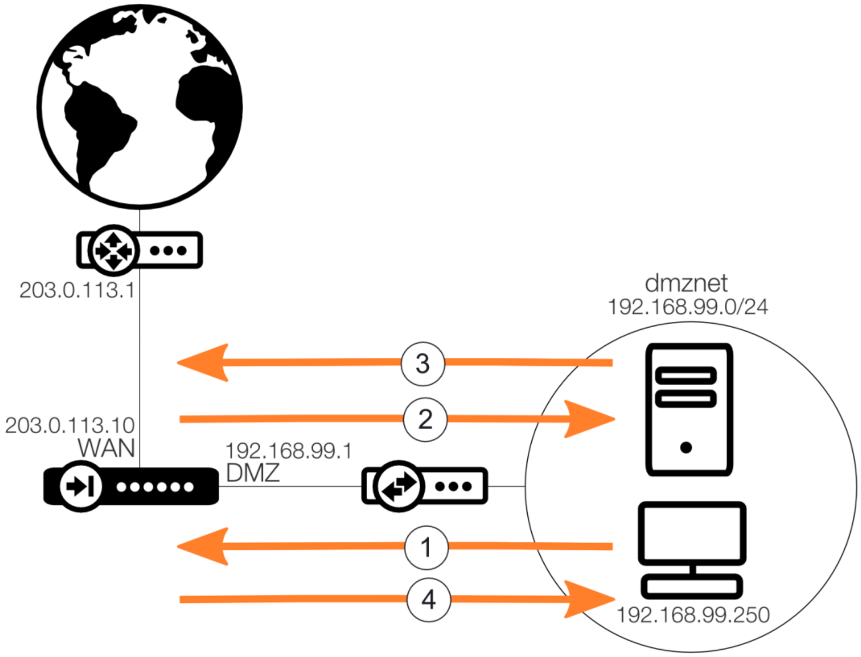


Figure 2.7.5 Traffic packet flows when using NAT

The event flow is now as follows:

- The client connects to the public IP address by typing in the web server’s public FQDN name (**1**).
- IP policy #2 trigger and forwards the client request to the server in DMZ (**2**).
- The server in DMZ gets a request from IP 192.168.99.1, which is part of the web server’s own network range (192.168.99.0/24).
- The web server (192.168.99.100) sends the reply to 192.168.99.1 (**3**) which is an IP belonging to cOS Core itself. The server does not reply to the client directly, avoiding the previous problem.
- cOS Core tracks the connections, so once the reply comes from the server, cOS Core will know where to return or forward the reply.
- The connecting client gets a reply from the IP it sent the request to and the user can access the web server (**4**).

Recipe 2.8. Extending administrator access and structuring IP policies further

Objectives

This recipe will go into details about modifying and/or adding IP policies to the current IP rule set to provide the administrator with as much access as possible to the clients, servers and other equipment in the network shown below in *Figure 2.8.1*.

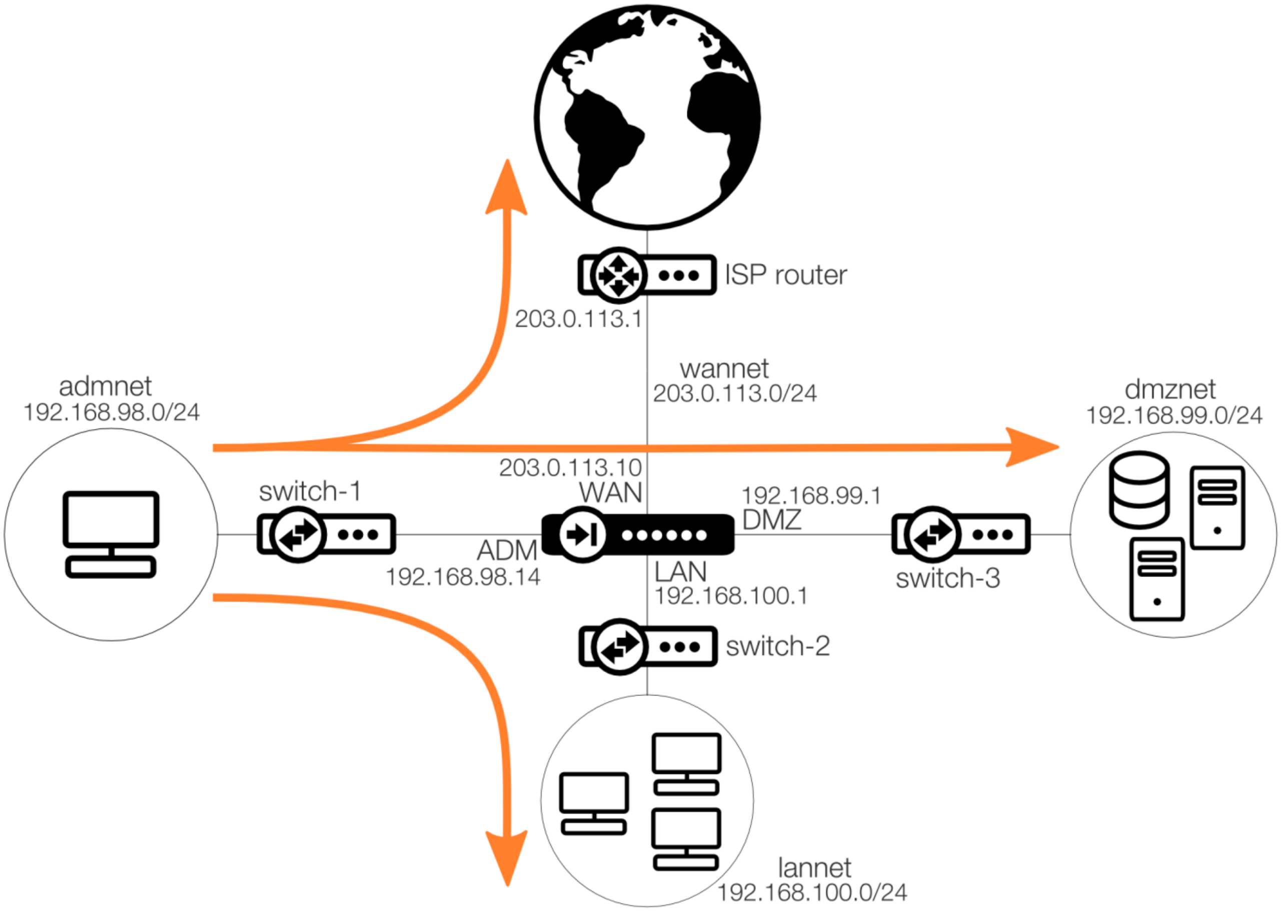
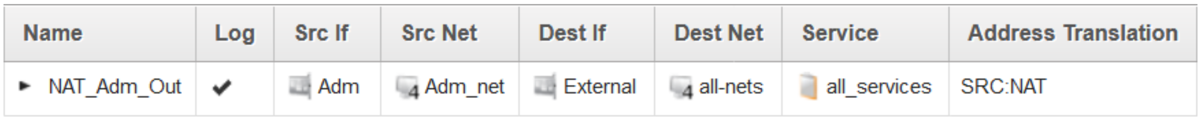


Figure 2.8.1 The complete basic network

Detailed Discussion

All communication between the various networks goes through cOS Core. We must make sure that the administrator has the access levels he needs to administrate the network. This includes access to web-servers, client PCs and all the switches and routers in the network.

Based on previous recipes, the current policy for administrator access looks like the one shown in the screenshot below.

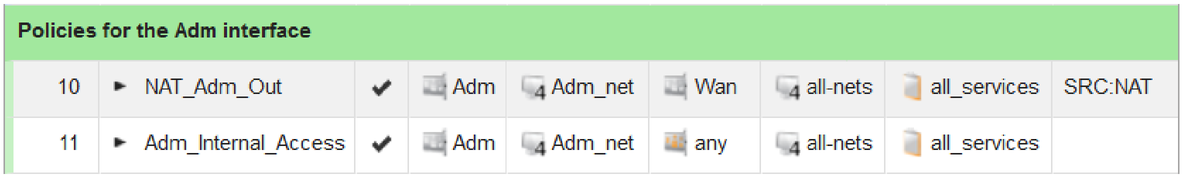


Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
► NAT_Adm_Out	✓	Adm	Adm_net	External	all-nets	all_services	SRC:NAT

Figure 2.8.2 The IP policy for the administrator

This IP policy means that the administrator currently only has Internet access, he cannot connect to anything in either the LAN or DMZ zones.

Administrators usually needs full access to everything on every port and protocol so we must add a second policy that provides access to every internal resource, both current and future. This policy is shown below.



Policies for the Adm interface

10	► NAT_Adm_Out	✓	Adm	Adm_net	Wan	all-nets	all_services	SRC:NAT
11	► Adm_Internal_Access	✓	Adm	Adm_net	any	all-nets	all_services	

Figure 2.8.3 Adding administrator access to internal resources

This new policy does not have any restriction on the destination interface or network. It allows full access to anything from the ADM interface.

Note also that the new policy does not use address translation. There is no immediate need to address translate internal communication unless we run into a scenario such as the one described in the previous recipe.

Of course there will always be exceptions and special circumstances that may require a NAT or even Stateless Policies but this book does not attempt to discuss all the possible scenarios.

Note

Giving the administrator interface and network complete access to all interfaces and all networks can be considered a security risk. All one would need at this stage would be to place a PC in the administration network and it would have full access to all network resources making a network scan extremely easy.

It will be up to the administrator to decide how much he wants to lock down access to the networks from the various interfaces. There are many security mechanisms and ways to configure cOS Core that can be implemented to lessen the impact of unauthorized access. Many of these will be discussed as we progress through the book.

Extending administrator access to the DMZ

Like other interfaces, we have not added the ADM interface to our interface group that is used for the incoming connection to the web server. After adding the ADM interface to our interface group it now looks as shown in the next screenshot.

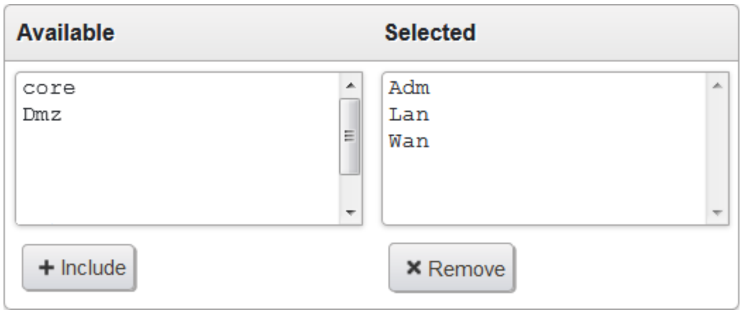
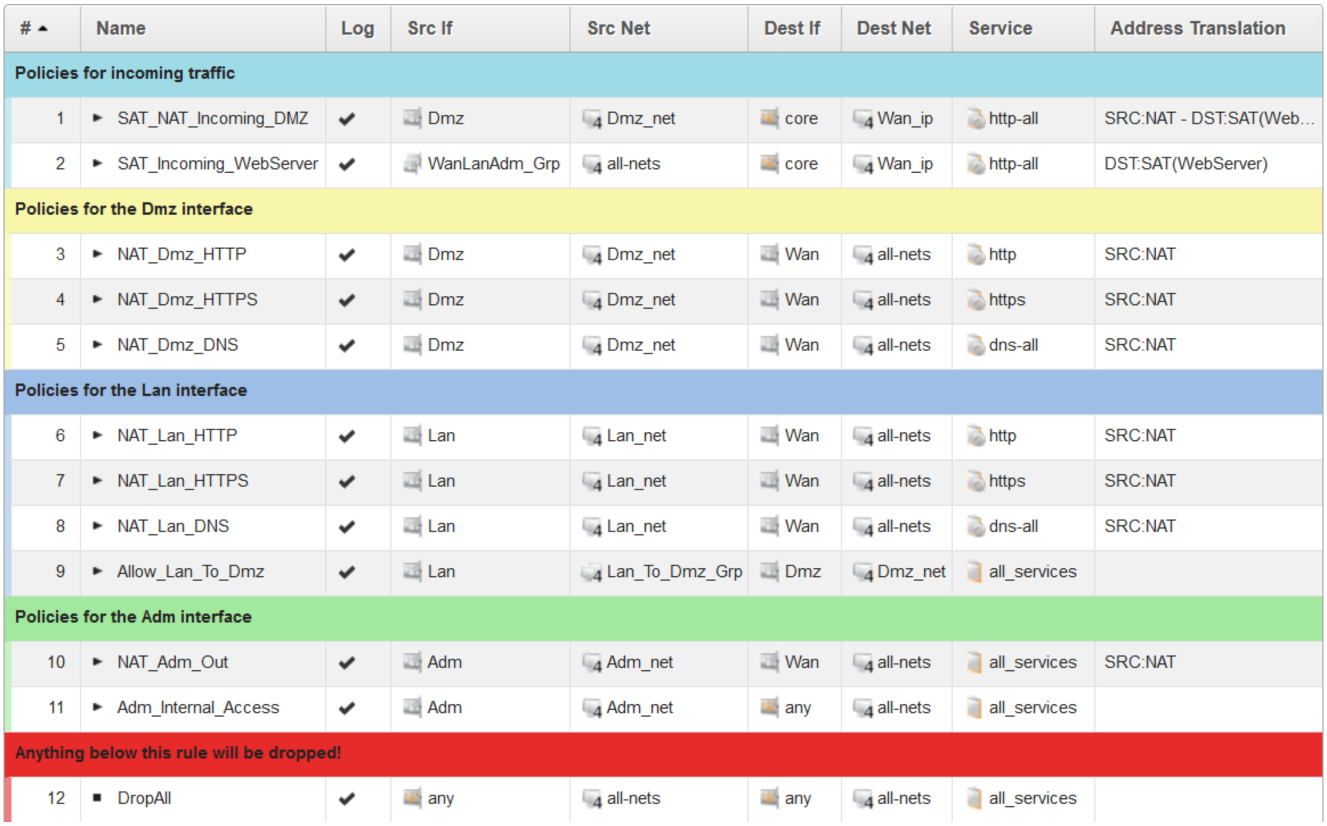


Figure 2.8.4 Adding the ADM interface to an interface group

Modifying our IP policies accordingly, the final version of the rule set built up in this chapter now looks as shown in Figure 2.8.5.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
Policies for incoming traffic								
1	► SAT_NAT_Incoming_DMZ	✓	Dmz	Dmz_net	core	Wan_ip	http-all	SRC:NAT - DST:SAT(Web...
2	► SAT_Incoming_WebServer	✓	WanLanAdm_Grp	all-nets	core	Wan_ip	http-all	DST:SAT(WebServer)
Policies for the Dmz interface								
3	► NAT_Dmz_HTTP	✓	Dmz	Dmz_net	Wan	all-nets	http	SRC:NAT
4	► NAT_Dmz_HTTPS	✓	Dmz	Dmz_net	Wan	all-nets	https	SRC:NAT
5	► NAT_Dmz_DNS	✓	Dmz	Dmz_net	Wan	all-nets	dns-all	SRC:NAT
Policies for the Lan interface								
6	► NAT_Lan_HTTP	✓	Lan	Lan_net	Wan	all-nets	http	SRC:NAT
7	► NAT_Lan_HTTPS	✓	Lan	Lan_net	Wan	all-nets	https	SRC:NAT
8	► NAT_Lan_DNS	✓	Lan	Lan_net	Wan	all-nets	dns-all	SRC:NAT
9	► Allow_Lan_To_Dmz	✓	Lan	Lan_To_Dmz_Grp	Dmz	Dmz_net	all_services	
Policies for the Adm interface								
10	► NAT_Adm_Out	✓	Adm	Adm_net	Wan	all-nets	all_services	SRC:NAT
11	► Adm_Internal_Access	✓	Adm	Adm_net	any	all-nets	all_services	
Anything below this rule will be dropped!								
12	■ DropAll	✓	any	all-nets	any	all-nets	all_services	

Figure 2.8.5 The final IP rule set for the chapter

Most scenarios involve access from only one or two interfaces. The more we restrict access to a particular resource the better. Configuring the use of Any or all-nets as interface or network should be done with special care.

It is always better to specify the narrowest access possible in IP policies. Not doing this could introduce "leaks" where the policies could allow unintended access to interfaces and networks.

Recipe 2.9. Adding a DHCP server on the LAN interface

Objectives

This recipe will discuss how to configure a DHCP (Dynamic Host Configuration Protocol) server on the LAN interface.

DHCP servers are extremely useful in any network no matter how many users are involved. They can hand out IP addresses and network definitions to clients automatically. Direct administrator intervention using static IP and network configuration is not needed. DHCP also helps to avoid accidentally allocating the same IP more than once. IP address duplication can introduce IP conflicts into a network which can cause many problems and can be difficult to troubleshoot.

Figure 2.9.1 below shows the LAN interface with a large number of connected clients which will require IP addresses allocated to them using DHCP.

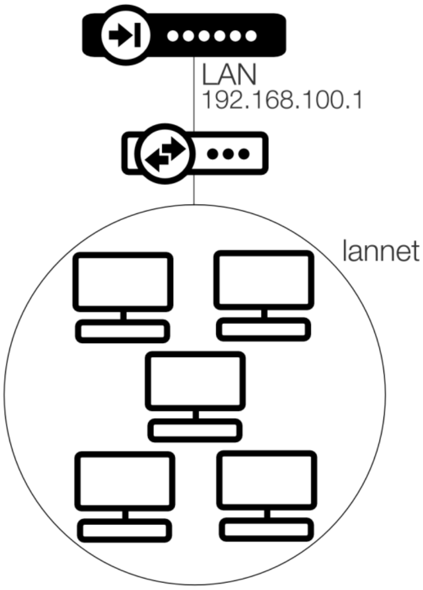


Figure 2.9.1 Multiple client PCs behind the LAN interface

Detailed Discussion

DHCP servers are added and configured in the WebUI by going to **Network > Network Services > DHCP > DHCP Servers**, as shown in the WebUI screenshot below.

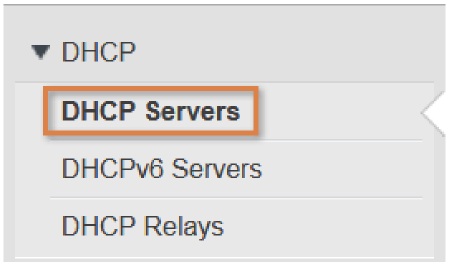
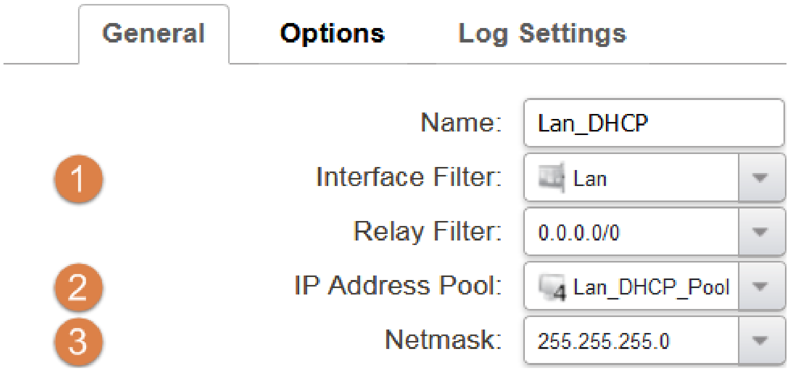


Figure 2.9.2 DHCP servers

When we create the DHCP server, we need to decide how many IP addresses the address pool should contain.

The network size on the LAN interface is a /24 network, so we have a total of 254 addresses to use. 192.168.100.1 is used by cOS Core itself so that address always has to be excluded. Depending on the amount of users in the network it is recommended to start at the upper levels of the network range in order to avoid conflicts with any statically set IP addresses as they are commonly set at the lower range.

We will therefore use a network range definition of 192.168.100.100-192.168.254, we can then have a maximum of 155 connected clients not counting the statically configured clients. The main DHCP server options are shown in the next screenshot.



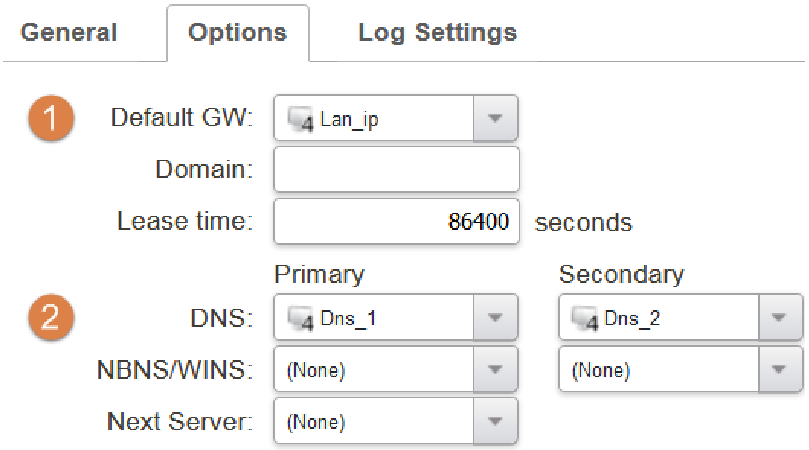
Name: Lan_DHCP
Interface Filter: Lan
Relay Filter: 0.0.0.0/0
IP Address Pool: Lan_DHCP_Pool
Netmask: 255.255.255.0

Figure 2.9.3 DHCP server general settings

First we select the interface filter (**1**) to be LAN. This is the interface the DHCP server will listen on for incoming DHCP requests.

Secondly, for the IP address Pool (**2**), we will use our previously defined IP pool network object. Lastly, we need to define a Netmask (**3**). As we have a network size of /24 we leave the netmask as the default of 255.255.255.0. This needs to be manually set when the network size is not the default /24 since the netmask calculation is not done automatically.

Next, we go to the **Options** tab. There are two more options that we must define in order to make the DHCP server fully functioning and these are shown in the next screenshot.



Default GW: Lan_ip

Domain:

Lease time: 86400 seconds

	Primary	Secondary
DNS:	Dns_1	Dns_2
NBNS/WINS:	(None)	(None)
Next Server:	(None)	

Figure 2.9.4 DHCP server options

The first option is the default gateway (**1**). This is the default gateway we will tell clients to use once they receive an IP lease offer from cOS Core. Since clients are behind the LAN interface and we want them to use cOS Core to reach the Internet, we select the Lan_ip interface for this.

Next we need to define a primary (**2**) and optionally a secondary DNS server. This is so that clients can make DNS queries without having to manually configure a DNS server on each client.

Create two new network objects in the cOS Core address book for the primary and secondary DNS server based on the information we got from our service provider, use these two network objects with the DHCP server.

After deploying these configuration changes we now have a fully functioning DHCP server.

Note

There are many options on the DHCP server we have not discussed such as the relay filter, the domain and lease times. The reason for this is because this is a basic setup. The options and settings discussed here are the bare minimum required to get a DHCP server up and running for the most common situations.

Recipe 2.10. Adding a log receiver located behind the DMZ interface

Objectives

This recipe discusses how to make cOS Core send logs to a target log receiver. cOS Core is constantly generating log event messages which record what is happening in the network. Messages can be anything ranging from DHCP server events to client connections being set up when connecting to a webpage on the Internet.

It is highly recommended to configure a log receiver and to look at the received log events when troubleshooting problems or looking at network usage. *Figure 2.10.1* below, illustrates a log receiver located behind the DMZ interface.

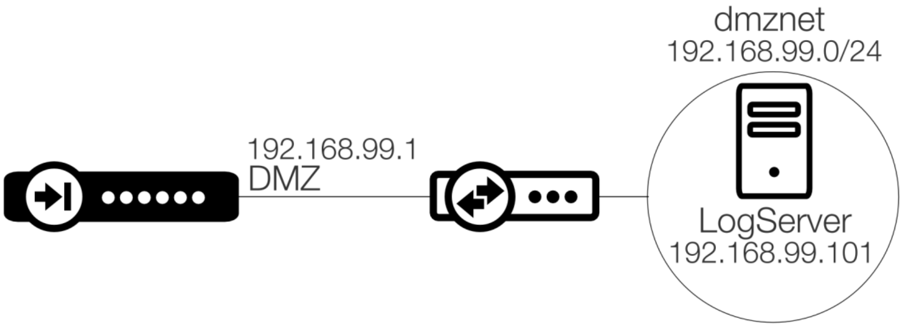


Figure 2.10.1 Syslog server located behind the DMZ interface

Detailed Discussion

The event messages generated by cOS Core can be sent to different types of log receivers. To receive messages, it is necessary to configure one or more event receivers objects in cOS Core that specify what events to capture, and where to send them. It is rare to change the default log event settings on the log receiver as the default settings cover almost all situations.

A log receiver can be either Memlog (memory log receiver), Syslog, an InControl Log Receiver, Mail Alerting or an SNMP (Simple Network Management Protocol) Event Receiver (traps) as shown in Figure 2.10.2.

A log receiver can be configured and/or added by going to **System > Device > Log and Event Receivers** in the WebUI.

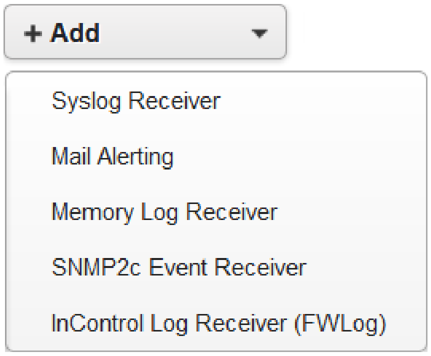


Figure 2.10.2 Adding a log receiver

Note

There can only be one Memory Log Receiver. If one is already configure the option to add another will not be present.

The different types of log receiver are:

- **MemoryLogReceiver**

  cOS Core has its own logging mechanism also known as the MemLog. This retains a limited number of log event messages in memory and allows direct viewing of recent log messages through the Web Interface. As this is only stored in memory it will quickly be overwritten with new log entries as they are generated. The memory log will also be reset when the system restarts.

- **Syslog Receiver**

  Syslog is the de-facto log message standard for logging events from network devices.

- **InControl Log Receiver**

  The separate Clavister Centralized Management product has the ability to receive and analyze log event messages for one or many Clavister Next Generation Firewalls. Event messages sent to the InControl log receiver use a Clavister proprietary event message format for logging called FWLog. This format has a high level of detail and is suitable for allowing analysis of large amounts of log data.

- **SNMP Traps**

  An SNMP2c Event Receiver can be defined to collect SNMP Trap log messages. These receivers are typically used to collect and respond to critical alerts from network devices.

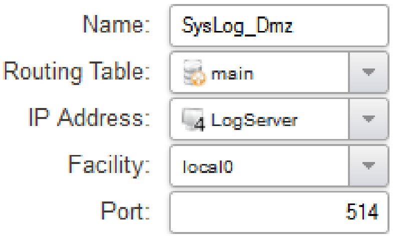
- **Mail Alerting**

  Mail Alerting can be configured to send log messages in an email to a specified email address via a nominated external SMTP server.

  The intended purpose of this feature is to provide a means of quickly alerting the administrator of any important cOS Core events so the selected level of severity for the events sent in this way will usually be very high.

Adding a syslog receiver

As an example, we will add a syslog receiver located behind the DMZ interface. For a basic setup the only thing we need to do in cOS Core is to create a network object with the IP to the log receiver server (192.168.99.101 in this case) and then select that IP address object in the SysLog receiver configuration, as shown below.



Name: SysLog_Dmz
Routing Table: main
IP Address: LogServer
Facility: local0
Port: 514

Figure 2.10.3 Configuring a Syslog receiver

Once the configuration is deployed, cOS Core will start to send any log event generated to the log receiver. It is also possible to exclude some log categories and log IDs sent to the log receivers if they are of no interest to the administrator. Keep in mind, however, that some log events are needed to compile reports such as for bandwidth usage. If critical log entries are not included, the report will be incomplete or will not display the desired data at all.

The default setting in cOS Core limits the maximum amount of logs sent per second to 2000, but it can be increased if the need arises.

This concludes the basic setup chapter.

Chapter 3: The University

3.1. Introduction

Welcome to the Clavister University campus. In this chapter we will consider a situation where we want to design a university network from the ground up. Going from a basic setup to a more advanced network, implementing more advanced functionality such as Web Content Filtering, Anti-Virus, Application Control, bandwidth management and more.

From this chapter and beyond, we will assume you are familiar the Clavister way of configuring policies, routing principles and so on.

Some of the basic IP policies, management access and routes have already be implemented earlier in this book. If you have not read the previous two chapters it is recommended that you do so if you have no prior experience with Clavister and cOS Core.

We will start with an outline of what we want to implement. The basic idea is that every student should have Internet access on campus and in their dormitory, by using either Wi-Fi or a normal Ethernet connection.

Figure 3.1.1 The university campus

Further into this chapter, we will implement restrictions on bandwidth and what students are allowed to access on the Internet in terms of webpage classifications.

The network will be built in a similar way to the previous chapter, where we start with a simple network and gradually add more and more functionality, features and complexity to the configuration. The recipes can be used individually but there may be references to the university network and its setup.

3.2. Setting Up the Network

Before we start configuring the various recipes we need to discuss an overview of the network. We need some initial network definitions and network objects similar to the basic setup discussed in the previous chapter.

An overview of the campus network is shown below in *Figure 3.2.1*.

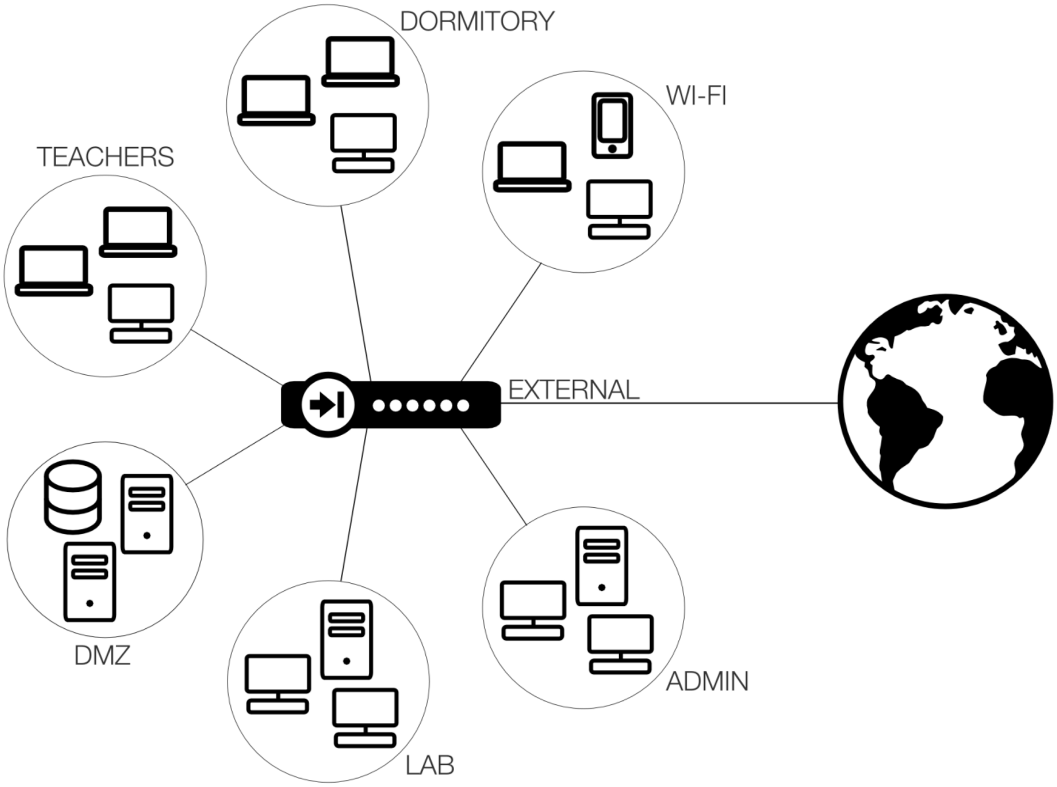


Figure 3.2.1 University campus network overview

We have chosen a Clavister Next Generation Firewall with 8 physical Ethernet interfaces for this installation.

For now, the initial address book definitions for these interfaces are listed below in *Table 3.2.2*.

Interface Name	Network
EXTERNAL	203.0.113.0/24
Wi-Fi	10.10.0.0/16
DORMITORY	10.20.0.0/16
TEACHERS	172.16.0.0/24
DMZ	172.16.10.0/24
H1	172.16.20.0/24
LAB	192.168.0.0/24
ADMIN	192.168.254.0/24

Table 3.2.2 Address book definitions

We have configured IP addresses and networks on all interfaces, even if they may not be used straight away. When configuring cOS Core for the first time each interface (except the management interface) has an IP address and network in the localhost 127.x.x.x IP range.

As shown in the addresses above, we have defined the network on the Wi-Fi and DORMITORY interfaces as /16 sized networks instead of the more usual /24. The reason for this is that there will be a need for large amounts of IP addresses on these interfaces.

By extending the network size to be /16 we gain access to more than 65,000 address compared to the standard 254 for a /24 sized network. We expect to have a large number of users and devices on these interfaces.

But what about the LAB interface? If we have hundreds and maybe thousands of students who at some point may need to use the lab environment, will 254 addresses be enough?

No they won't, but we will also handle this situation with a different approach in a later chapter by using VLANs and Virtual Routing.

Tip

By setting unique networks and IP addresses on each interface, it will make things much easier when troubleshooting potential problems in the network.

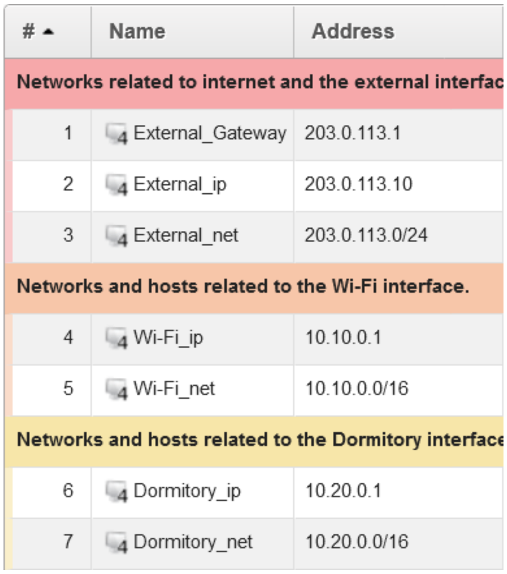
If you see a particular network that is not supposed to be there, you immediately know where this network is routed and subsequently what to look for in the configuration or connected switches/routers.

Seeing multiple requests from 127.0.0.1 will make it that much more difficult to troubleshoot potential problems.

As mentioned in the first chapter, it is highly recommended that comment groups are used in the majority of the configuration.

When dealing with lots of objects, interfaces and routes, it will be much easier to get an overview of the configuration if everything is well documented.

An example of using comment groups in the cOS Core Address Book is shown in the next WebUI screenshot.



# ▲	Name	Address
Networks related to internet and the external interfac		
1	External_Gateway	203.0.113.1
2	External_ip	203.0.113.10
3	External_net	203.0.113.0/24
Networks and hosts related to the Wi-Fi interface.		
4	Wi-Fi_ip	10.10.0.1
5	Wi-Fi_net	10.10.0.0/16
Networks and hosts related to the Dormitory interface		
6	Dormitory_ip	10.20.0.1
7	Dormitory_net	10.20.0.0/16

Figure 3.2.3 Comment group usage

The intended use for each interface of our Clavister firewall is summarized next in *Table 3.2.4*.

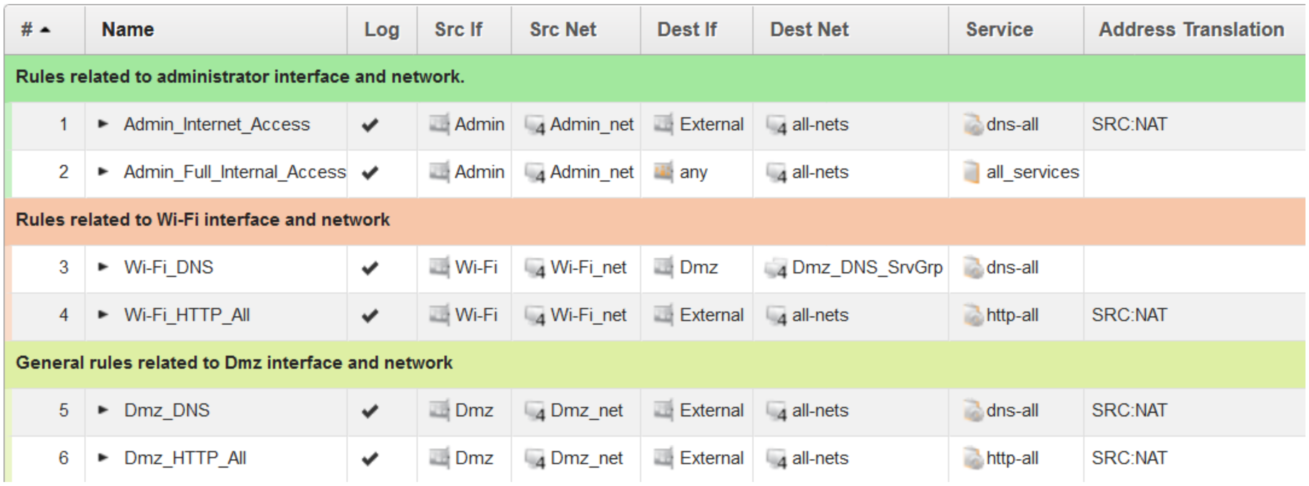
Interface Name	Usage
Wi-Fi	Campus Wi-Fi access. Everyone will have access to the Wi-Fi network as long as they are a part of the university.
DORMITORY	The physical Ethernet connection to each student room will be going through this interface. Less restricted than the Wi-Fi network.
TEACHERS	The physical Ethernet connection to each teacher computer in teacher offices as well as classrooms will be going through this interface. Less restricted than both the DORMITORY and Wi-Fi interfaces.
DMZ	The university DMZ zone. Location of all web servers, FTP servers, storage devices etc. limited access for everyone except the administrators.
H1	Reserved interface for use in High Availability
LAB	Special lab network that has unrestricted access to the Internet. Heavily segmented using VLAN to avoid that different lab sections interfere with each other.
ADMIN	The physical Ethernet connection for the administrators will be going through this interface. Unlimited access to all aspects of the university network.

Table 3.2.4 Interface usage

Currently Configured Interface IP Policies

Each interface currently has two IP policies configured, one to allow DNS requests to the DNS servers located in the DMZ and another to allow HTTP and HTTPS towards the Internet. Any internal access is currently denied. The exception is the administrators, who have full access to both internal and external networks.

A small sample of the IP rule set is shown in the next screenshot from the WebUI.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
Rules related to administrator interface and network.								
1	► Admin_Internet_Access	✔	Admin	Admin_net	External	all-nets	dns-all	SRC:NAT
2	► Admin_Full_Internal_Access	✔	Admin	Admin_net	any	all-nets	all_services	
Rules related to Wi-Fi interface and network								
3	► Wi-Fi_DNS	✔	Wi-Fi	Wi-Fi_net	Dmz	Dmz_DNS_SrvGrp	dns-all	
4	► Wi-Fi_HTTP_All	✔	Wi-Fi	Wi-Fi_net	External	all-nets	http-all	SRC:NAT
General rules related to Dmz interface and network								
5	► Dmz_DNS	✔	Dmz	Dmz_net	External	all-nets	dns-all	SRC:NAT
6	► Dmz_HTTP_All	✔	Dmz	Dmz_net	External	all-nets	http-all	SRC:NAT

Figure 3.2.5 Current policies for the ADMIN, DMZ and Wi-Fi interfaces

Observant readers may note that the DNS policy for DMZ looks different. This is because we need the DNS servers located here to be able to reach other DNS servers on the Internet in order to fetch and store external DNS records. Otherwise, users using the DNS server in the DMZ zone will not be able to resolve external addresses on the Internet.

Other servers located on the DMZ interface that use the DMZ DNS servers do not require any special policy. That traffic will not pass through the firewall since they are located on the same local network segment and these requests will simply go through the local switch.

It depends on network conditions of course, so the policy requirements may vary.

Recipe 3.3. Configuring DHCP

Objectives

This recipe will discuss the configuration of DHCP (Dynamic Host Configuration Protocol) servers on the majority of the interfaces used in the university. DHCP servers were briefly discussed in *Recipe 2.9. Adding a DHCP server on the LAN interface*. In this recipe, we will configure and use DHCP servers in our university environment.

Figure 3.3.1 When many users require an IP address, DHCP is needed

Detailed Discussion

DHCP servers are used to automatically hand out IP addresses to connected clients without the need to manually configure the address, gateway or DNS server(s). This is needed in many scenarios due to the large amount of work that otherwise would have to be done in manually configuring each client.

This is especially true in a University, which can contain thousands of PCs, servers, phones and other devices that need their own IP address.

To configure a DHCP server we need to create a minimum of 3 address book objects:

1. An IP address pool. This is a range of IP addresses that DHCP can hand out to the clients.

2. A gateway address. This is used by the clients as their default gateway once they get an IP lease from the DHCP server.

3. One or more DNS server IP addresses. This is handed out to clients at the same time as the client IP address and gateway address and tells them which server(s) they should contact in order to make DNS queries.

Configuring the DHCP Objects

We define the following DHCP pools for the various interfaces and networks, as shown below in *Table 3.3.2*.

Name	Interface	DHCP Pool
Wi-Fi DHCP Pool	Wi-Fi	10.10.128.0/17
DORMITORY DHCP Pool	DORMITORY	10.20.128.0/17
TEACHERS DHCP Pool	TEACHERS	172.16.0.21-172.16.0.254
DMZ DHCP Pool	DMZ	172.16.10.201-172.16.10.254
LAB DHCP Pool	LAB	192.168.0.101-192.168.0.254
ADMIN DHCP Pool	ADMIN	192.168.254.128/25

Table 3.3.2 DHCP IP Pools allocated to various interfaces

Now, let's look at why we defined these pool objects as we did.

Our choices all come down to what kind of interface it is and what we expect to be behind this interface. If we start by looking at the Wi-Fi and DORMITORY DHCP pool, we have split the entire /16 network in half and allocated the upper portion of the network to the DHCP pool, which means more than 32,000 addresses. We could naturally use the entire /16 as well but initially there is no need to have such a huge IP pool. We therefore leave more than enough room for future network modifications and if the need for IPs increases, the IP pool can be easily extended to include a larger portion of the /16 network.

The teacher's IP pool contains 234 IP addresses. We leave the first 20 in case we need an increase and if we will assign some users static IP addresses.

The same goes for the DMZ interface, but here we have different requirements in the network. The DMZ network will contain many servers and the majority of those servers will have static IP addresses. So the need to have a DHCP server on the DMZ interface is still valid but its IP pool will be significantly smaller, in this case only 54 IP addresses.

The same again for the lab network. Since most testing equipment has configured IP addresses we have a similarly small size for its DHCP server address pool.

And lastly, the ADMIN interface. Here we expect a mix of static and dynamic IP assignment, so we simply split the network in half using a /25 netmask.

The Gateway and DNS objects

The gateway address object in most cases already exists when we come to this stage. If we have a user behind the Wi-Fi network, the default gateway for such a user will be the IP address used by the Wi-Fi interface. And in our university we have already defined that object earlier.

For DNS, we create two address book objects that point to the DNS servers that we want our clients to use once they get a lease from the DHCP server. Our two university DNS servers are located behind the DMZ interface.

Configuring the DHCP server

Now, when the objects needed are defined and explained we will add and configure the DHCP servers itself. This is done under **Network > Network Services > DHCP servers** in the WebUI, as shown below.

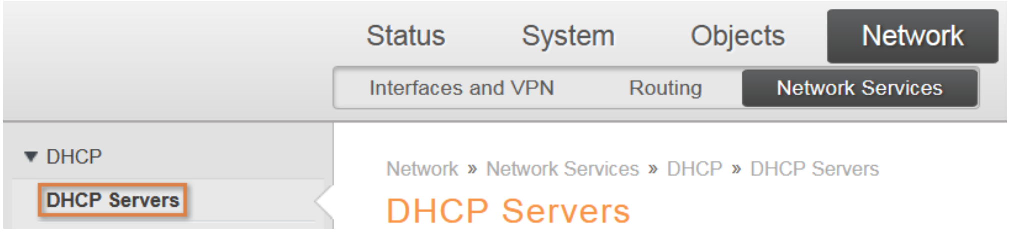
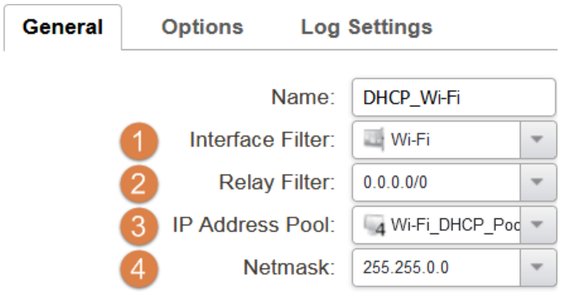


Figure 3.3.3 Defining a DHCP server in the WebUI

When the DHCP server is created, we need to define some of the standard options and settings. These are shown in the next screenshot from the cOS Core WebUI.



Name: DHCP_Wi-Fi
1 Interface Filter: Wi-Fi
2 Relay Filter: 0.0.0.0/0
3 IP Address Pool: Wi-Fi_DHCP_Poc
4 Netmask: 255.255.0.0

Figure 3.3.4 Setting general DHCP server options

First (besides the name), we need to choose which interface the DHCP server should listen on (**1**). We will use the Wi-Fi interface in this example.

Next, is the Relay Filter (**2**). This can be used to restrict DHCP requests from either only the local interface or from a specific DHCP relayer. As we want to allow everything on the Wi-Fi interface, we will leave it as the default value of 0.0.0.0/0 (in other words, all-nets).

The next option (**3**) defines the IP address that should be handed out to the requesting client. Here, we will use the Wi-Fi pool object we created earlier.

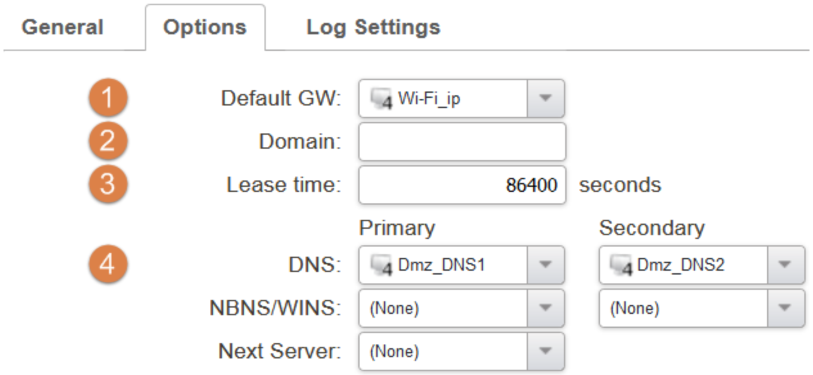
Lastly is the Netmask (**4**) that will be forwarded to the requesting client in the lease offer.

Make sure that the netmask is set correctly depending on the size of the network used. The pre-selected netmask value is 255.255.255.0 which is a class C network, but in the case of the Wi-Fi and DORMITORY network it will be a class B network, so the netmask will be 255.255.0.0.

Note

The netmask value on the DHCP server is not based on the size of the DHCP pool. It is based on the size of the network.

Next, we will select the **Options** tab, as shown in the screenshot below.



Default GW: Wi-Fi_ip

Domain:

Lease time: 86400 seconds

	Primary	Secondary
DNS:	Dmz_DNS1	Dmz_DNS2
NBNS/WINS:	(None)	(None)
Next Server:	(None)	

Figure 3.3.5 Setting other DHCP server options

Here, the default gateway (**1**) is the Clavister firewall's Wi-Fi interface IP address. Any client that gets an IP from the DHCP server will automatically use this as their default gateway for networks outside its own network segment, which in this particular case, will be anything not part of the Wi-Fi 10.10.0.0/16 network.

The domain option (**2**) is the domain name used for domain suffix. The idea of a domain name is that the client will know which domain it belongs to, so when you type in just the word "test" into the web browser it will try to connect to "test.example.com" because it knows the domain name. We will leave this option blank for the moment but once the university domain is up and running this will most likely be something that will need a value.

The lease time (**3**) is the time, in seconds, that a DHCP lease is provided. After this time the DHCP client must renew the lease. We will leave this setting at its default value of 86400 seconds (24 hours).

Next, we define two DNS servers (**4**) located in our DMZ.

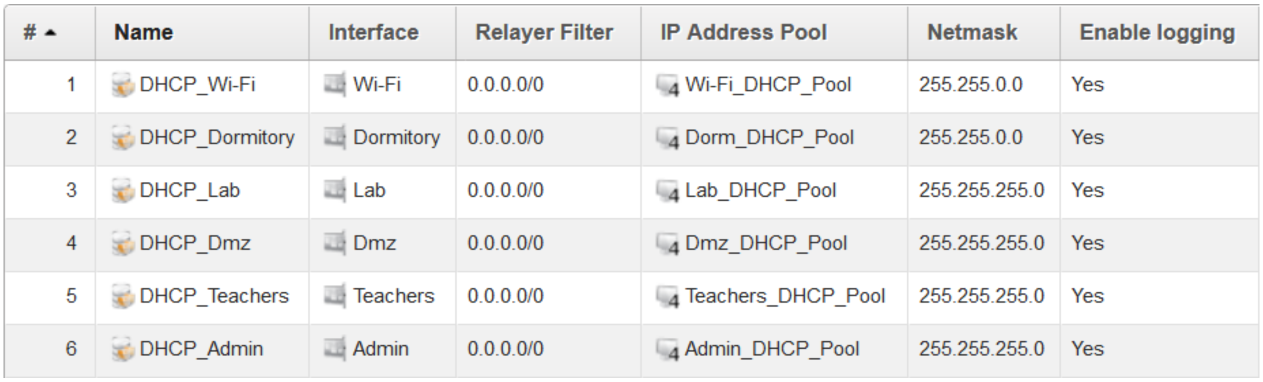
Lastly, a short note explaining the NBNS/WINS and Next Server options:

- **NBNS/WINS** is the IP of the Windows Internet Name Service (WINS) servers that are used in Microsoft environments which uses the NetBIOS Name Servers (NBNS) to assign IP addresses to NetBIOS names.

- **Next Server** Specifies the IP address of the next server in the boot process. This is usually a TFTP server.

Neither of the above two options will be used in this book.

Now, we have a fully functioning DHCP server and we can now proceed to add DHCP servers for the other interfaces and networks, as shown in the screenshot below.



# ▲	Name	Interface	Relayer Filter	IP Address Pool	Netmask	Enable logging
1	DHCP_Wi-Fi	Wi-Fi	0.0.0.0/0	Wi-Fi_DHCP_Pool	255.255.0.0	Yes
2	DHCP_Dormitory	Dormitory	0.0.0.0/0	Dorm_DHCP_Pool	255.255.0.0	Yes
3	DHCP_Lab	Lab	0.0.0.0/0	Lab_DHCP_Pool	255.255.255.0	Yes
4	DHCP_Dmz	Dmz	0.0.0.0/0	Dmz_DHCP_Pool	255.255.255.0	Yes
5	DHCP_Teachers	Teachers	0.0.0.0/0	Teachers_DHCP_Pool	255.255.255.0	Yes
6	DHCP_Admin	Admin	0.0.0.0/0	Admin_DHCP_Pool	255.255.255.0	Yes

Figure 3.3.6 DHCP server summary

Once again, don't forget to use the *clone* option for faster creation of similar objects.

Note

There is a behavior when editing settings on the DHCP server in the WebUI that some may find confusing.

If we open an already created DHCP server we will end up at ***Static Host*** *and* ***Custom Options****. In order to get back to correct view for editing, we need to click the* ***Edit this object*** *button, as shown below.*

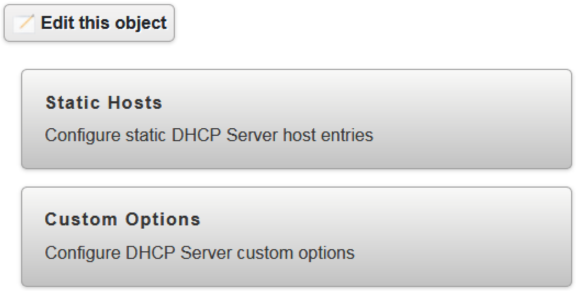

Edit this object
Static Hosts
Configure static DHCP Server host entries
Custom Options
Configure DHCP Server custom options

Figure 3.3.7 DHCP editing behavior

Recipe 3.4. Configuring a High Availability Cluster

Objectives

In our university example, the Clavister Next Generation Firewall stands as a central point in the network. If the firewall for one reason or another encounters some sort of problem, large portions of the network will go down. To avoid this happening we will implement a High Availability Cluster (HA cluster) to add additional redundancy in the network.

In this recipe, we will describe what a cluster is along with detailed description of all the various cluster settings and how HA is configured.

Detailed Discussion

The purpose of an HA cluster

cOS Core High Availability (HA) provides a fault tolerant capability to Clavister Next Generation Firewall installations. HA works by adding a back-up "slave" Clavister firewall to an existing "master" firewall. 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 cluster slave will be inactive and will only monitor the activity of the master. If the slave detects that the master has become inoperative, an HA failover takes place and the slave becomes active, assuming processing responsibility for all traffic.

If the master later becomes operative again, the slave will continue to be active but the master will now monitor the slave, with failover only taking place if the slave fails. This is sometimes known as an active-passive implementation of fault tolerance.

Using a cluster has the key benefit that even if one of the firewalls in the cluster were to encounter a problem, the network as a whole is not affected. The physical configuration of a cluster requires some additional hardware to achieve this redundancy.

Figure 3.4.1 below shows a simple configuration using a standalone unit utilizing only two interfaces, one for internal clients and another for the external Internet access through a router.

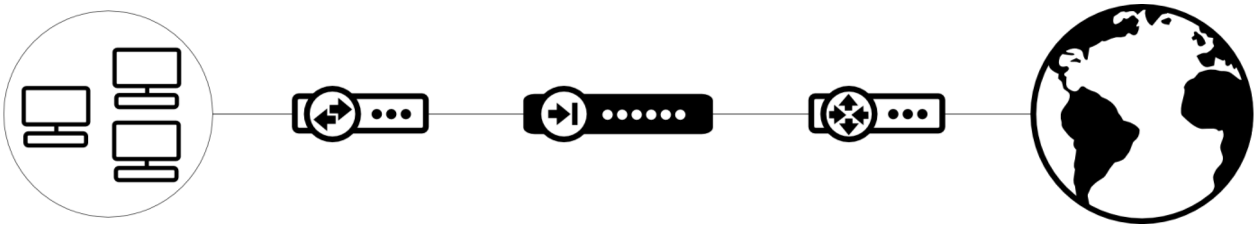


Figure 3.4.1 A standalone configuration

For an HA cluster, the configuration above will become slightly different. As shown below in *Figure 3.4.2*, both the master and slave unit must be connected to the same switch, this is because both units must be able to reach all aspects of the network no matter which node that is the active one.

This applies to the external interface as well, although it depends on what kind of router it is. If the router has a switch of its own or multiple ports there is no need to have a switch towards the router (but it is the most common scenario). The dotted line in *Figure 3.4.2* indicates the direct connection between the cluster nodes of the interface H1, which has been designated as the synchronization interface.

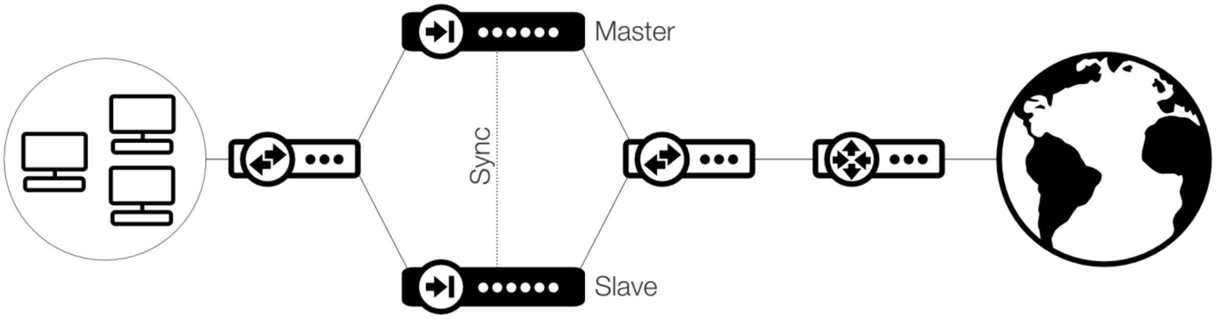


Figure 3.4.2 An HA cluster configuration

Note

It should be kept in mind that the master unit in a cOS Core cluster is not always the same as the active unit in a cluster.

Before we create the cluster itself we must prepare some new address book objects. A standalone unit only has one IP address per interface, but a cluster has 3. One IP will be used as the *Shared IP* and the other two will be the individual IPs used by each cluster node.

The *Shared IP* of an interface means the IP address both cluster nodes will use.

Advanced information about shared IPs 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 node. 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 cOS Core from the Cluster ID in the form 11-00-00-C1-4A-nn where nn is derived by combining the configured Cluster ID with the hardware bus/slot/port of the interface. The Cluster ID must be unique for each cOS Core cluster in a network.

As the shared IP address always has the same MAC 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 during 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 MAC address.

Continuing HA cluster setup

As mentioned previously, we need 3 IP addresses, one for each cluster interface. The Shared IP address we have already created as we will use each interface IP address as the shared. What we are missing is the individual IP addresses commonly referred to as Master_IP and Slave_IP for each interface.

These objects are of a special type that can be created in the address book, as shown below.

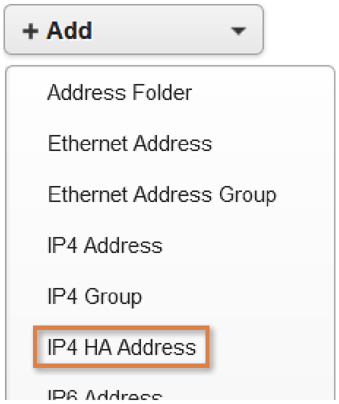
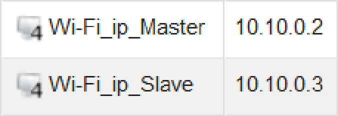


Figure 3.4.3 Creating IPv4HA cluster objects

Before we create these particular objects, we must first create two new normal IP addresses for each interface used in the cluster, as shown below.



Wi-Fi_ip_Master	10.10.0.2
Wi-Fi_ip_Slave	10.10.0.3

Figure 3.4.4 Creating master and slave IP address book objects

A common practice when creating a cluster is to use the first 3 addresses (or last) in the network range and reserve those for use by the cluster. As shown in figure above, for the Wi-Fi interface, we designate IP address .2 for the master and .3 for the slave.

All interfaces do not need three IP addresses

The individual HA addresses are primarily useful for the administrator. The two most important functions the individual addresses are used for are:

- **Management**

  Management must be done towards the individual address as the shared IP is a "virtual" IP shared between both cluster nodes.

- **Log Event Message Generation**

  When log messages are sent by cOS Core to a log receiver, it must have a unique IP address as sender. We cannot use the shared IP address as then we would have no way to know if it was the master or slave that sent them.

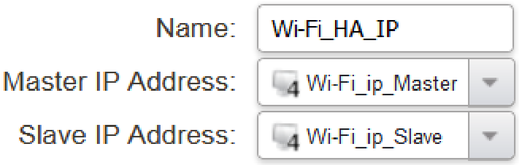
One concern when using clusters is the need to have 3 public IP addresses as that is not always possible. This is not a requirement unless we plan on managing the unit from the Internet and/or sending log messages directly to a host on the Internet.

We only need to set a public IP as the shared address and we can then use the auto-created address book object called "Localhost" as the master/slave IP. This applies to all other interfaces as well.

The default "Localhost" object consists of the address 127.0.0.1 and 127.0.02. Even though we can use these objects as HA objects on all interfaces (except management), it is recommended to set individual addresses anyway. When troubleshooting, seeing packets coming from 127.0.0.1 is troublesome when this address is set on multiple interfaces.

Assigning HA cluster objects to interfaces

In the university example configuration, we will designate 3 addresses for all interfaces. The objects we have created earlier will now be assigned to IPv4HA objects in the address book as shown in the next screenshot.

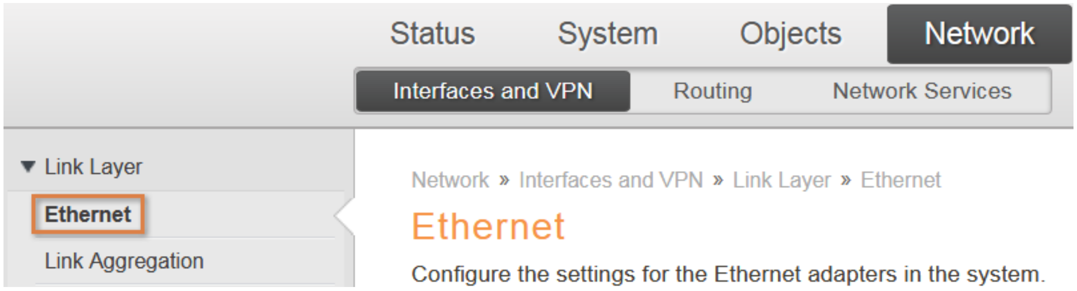


Name: Wi-Fi_HA_IP
Master IP Address: Wi-Fi_ip_Master
Slave IP Address: Wi-Fi_ip_Slave

Figure 3.4.5 Using IPv4 objects with an IPv4HA object

We repeat this procedure for all interfaces except the external interface as we have no need to either manage the cluster from the Internet or send logs to something on the Internet. Management and logs will be handled internally.

Once all objects and HA objects are created, we need to assign them to each interface. This operation can be done before the cluster itself is created. To do this we go to the Ethernet section in the WebUI, as shown in the following screenshot.



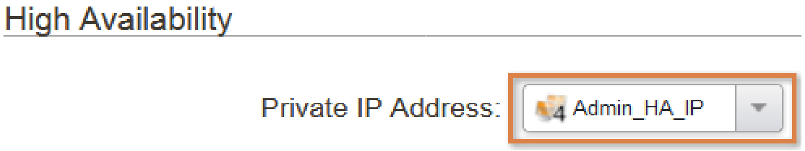
Network » Interfaces and VPN » Link Layer » Ethernet

Ethernet

Configure the settings for the Ethernet adapters in the system.

Figure 3.4.6 The Ethernet section in the WebUI

We open one of the interfaces then go to the advanced tab to locate the private IP address section, as shown in the next WebUI screenshot.



High Availability

Private IP Address: Admin_HA_IP

Figure 3.4.7 Setting a private IP address on each interface

We then set the correct Private IP address for each interface using the IPv4HA objects we created earlier, for the external interface we choose the default object "localhost".

Starting the HA Wizard

Now everything is ready to turn the configuration into a cluster. To accomplish this we go to the High Availability section in the WebUI as shown in the screenshot below.

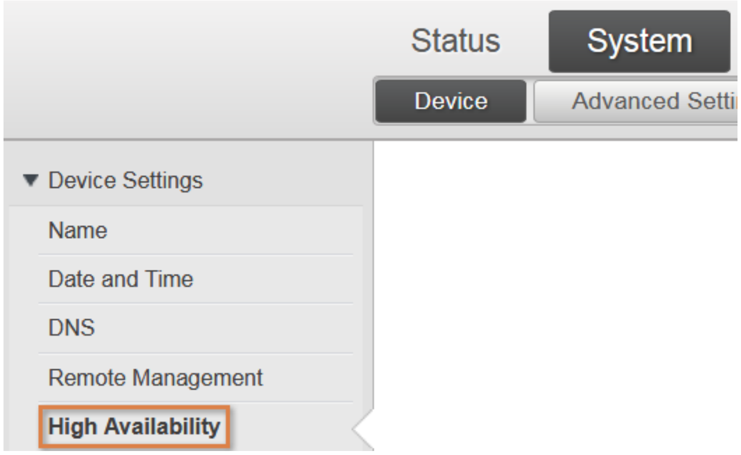
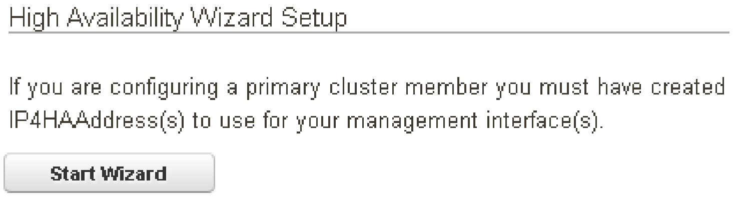


Figure 3.4.8 The HA WebUI location

Next, we want to initiate the High Availability cluster wizard as shown below.

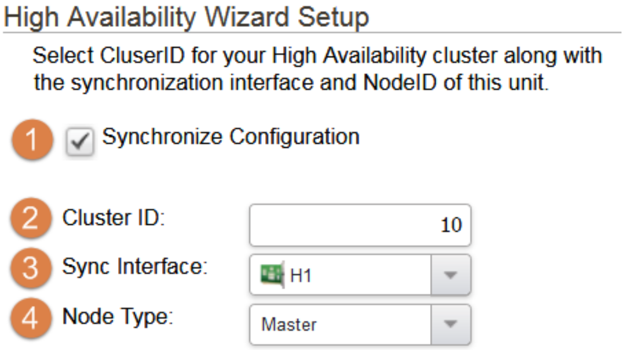


High Availability Wizard Setup

If you are configuring a primary cluster member you must have created IP4HAAddress(s) to use for your management interface(s).

Figure 3.4.9 Starting the HA wizard

The wizard requires us to have created IPv4 HA objects before we start it. Since we have already done that previously, we can proceed to use the HA Wizard and this is shown in the next screenshot.



High Availability Wizard Setup

Select CluserID for your High Availability cluster along with the synchronization interface and NodeID of this unit.

1 ☑ Synchronize Configuration

2 Cluster ID: 10

3 Sync Interface: H1

4 Node Type: Master

Figure 3.4.10 The first step using the HA wizard

HA wizard settings

Below is a description of the settings shown in the wizard screenshot above:

1. **Synchronize configuration** means that a configuration deployed on either of the master or slave automatically will be synchronized to the other node. Unchecking this checkbox means that we either have to perform individual configuration changes on the nodes or use Clavister InControl to synchronize the configuration.

2. This sets the Cluster ID. If this is not the only cluster in a network then the ID must be changed so that it is unique (the default value is 0). The ID is also used to generate the virtual MAC address used by the shared interface IPs. Since we do not know whether we will add any more clusters in the future, it is a good idea to move away from the default zero ID. In this case, the ID 10 is used. The cluster ID can be between 0 and 63.

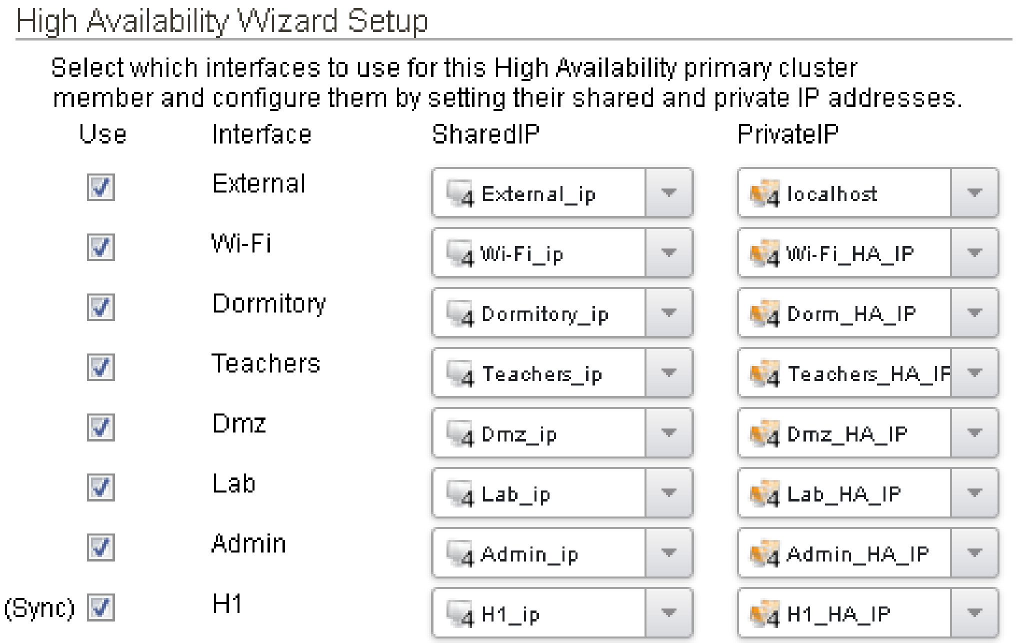
3. We can choose which interface we want to use as the synchronization interface. In a cluster, the master and slave units must be directly connected to each other by a synchronization connection which is known to cOS Core as the *sync* interface. One of the normal interfaces on the master and the slave are dedicated to this purpose and are connected together with a crossover cable. The cluster synchronizes data such as connections, IPsec tunnel states, route states and more.

This in order to cause as little traffic disturbances as possible if/when there is a cluster role change. An example would be when a configuration deployment is made, then at least one cluster role change will be performed. If everything is synchronized and up to date, normal users would not notice anything at all.

4. This determines what kind of node this unit is. There are two options, master or slave. The first unit configured in the cluster is usually designated as the master.

Finalizing the cluster for the master

In the final step of the HA Wizard setup for the master, as shown below, is a summary of all the interfaces, shared IPs and private IPs. Any last minute changes can be performed here before deploying and activating the cluster.



High Availability Wizard Setup

Select which interfaces to use for this High Availability primary cluster member and configure them by setting their shared and private IP addresses.

Use	Interface	SharedIP	PrivateIP
☑	External	External_ip	localhost
☑	Wi-Fi	Wi-Fi_ip	Wi-Fi_HA_IP
☑	Dormitory	Dormitory_ip	Dorm_HA_IP
☑	Teachers	Teachers_ip	Teachers_HA_IP
☑	Dmz	Dmz_ip	Dmz_HA_IP
☑	Lab	Lab_ip	Lab_HA_IP
☑	Admin	Admin_ip	Admin_HA_IP
(Sync) ☑	H1	H1_ip	H1_HA_IP

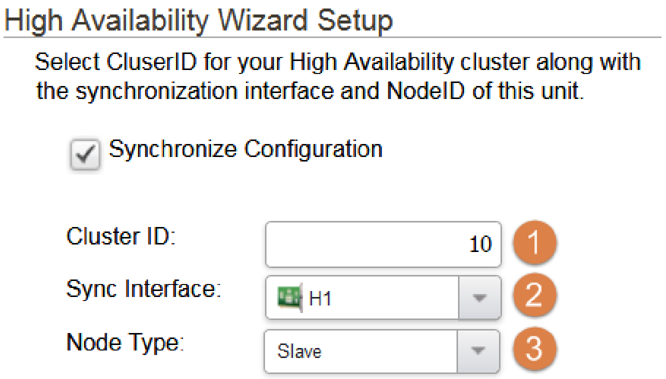
Figure 3.4.11 Summary of shared and private IPs on each interface

Once the configuration is deployed, the cluster is only 50% complete. It is a cluster which does not have a peer node. The peer node will be the slave.

Configuring the slave

After the master is configured and running, we need to add the slave node. The procedure to add the slave is slightly different but much easier than adding the master.

We connect to the slave unit's WebUI then go to the High Availability WebUI section and start the HA wizard again. Here we need to change three settings which are shown in the next screenshot.


High Availability Wizard Setup

Select ClusterID for your High Availability cluster along with the synchronization interface and NodeID of this unit.

☑ Synchronize Configuration

Cluster ID: 10

Sync Interface: H1

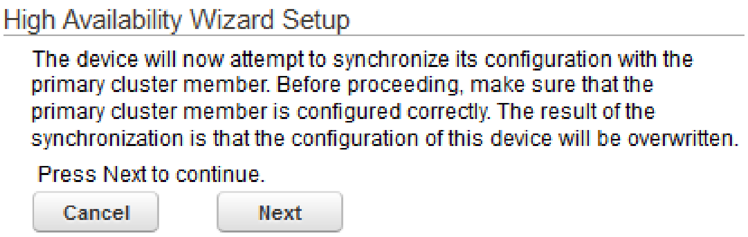
Node Type: Slave

Figure 3.4.12 The three slave settings needing modification

The settings are changed as follows:

1. We need to configure the Cluster ID to be the same as for the master unit.
2. We define the synchronization interface, again to be the same as on the master unit.
3. We change the Node Type from Master to Slave.

What this means is that when we continue to the next step in the HA wizard, the slave will attempt to contact the master and fetch the HA configuration from it. This requires that the synchronization interface and cluster ID are configured the same way, otherwise the master unit will reject all the connection attempts from the slave.


High Availability Wizard Setup

The device will now attempt to synchronize its configuration with the primary cluster member. Before proceeding, make sure that the primary cluster member is configured correctly. The result of the synchronization is that the configuration of this device will be overwritten.

Press Next to continue.

Figure 3.4.13 The HA wizard for the slave node

The last step is to verify the interface mappings. As the hardware is the same, we simply select the same interface mappings as the master.

Once the configuration is deployed, the HA cluster is now up and running.

Verifying that the cluster is working

To verify that the cluster is working as it should be, we can perform two tests:

- Deploy a small configuration change on either of the cluster node's, then verify that the change is present on the other cluster node as well.
- Connect to the CLI of either of the cluster nodes, then run either of the following CLI commands (depending on whether it is the active or inactive node).

  On the active node:

  ```
  Device:/> HA -deactivate
  ```

  Or on the inactive node:

  ```
  Device:/> HA -activate
  ```

If the cluster works as it should, both cluster nodes should change its active/inactive state. The active node should become inactive and vice versa.

Note

In the rare case of configuration synchronization not working when testing, try rebooting both cluster nodes. This should be a one-time operation.

An alternative way to add the slave

An alternative procedure to configure the cluster slave is to take a backup of the master's configuration and then restore it on the slave without activating it.

Before the configuration is activated, we change the node type from master to slave then deploy the configuration, as shown previously in figure 3.4.12.

Recipe 3.5. Restricting web access with Web Content Filtering

Objectives

The objective of this recipe is to impose restrictions on what kind of web categories are allowed through the Clavister firewall. By this we mean categories such as News, Gaming Sites, Business oriented, Adult Content, Chatrooms and so on. We do not, for instance, want to allow our university students to connect to pages classified as "Adult Content", as illustrated below in *Figure 3.5.1*.

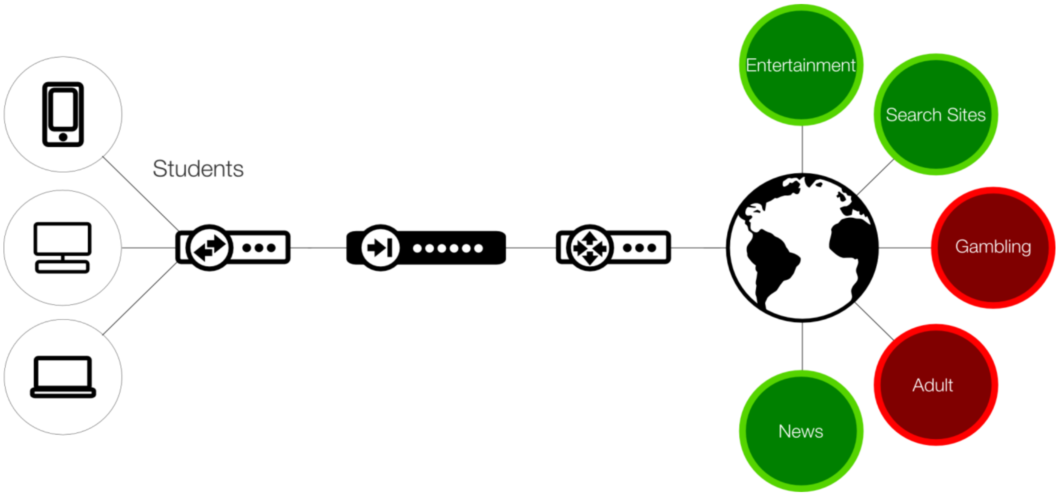


Figure 3.5.1 Blocking web categories for students

Note

Even though the majority of this chapter is based on using an HA cluster, we will use network schematics that do not show the cluster for easier visualization (and to avoid the need for large schematics).

Detailed Discussion

Difference between using ALGs in IP Policies and in IP Rules

Web content filtering (WCF) is a part of something called an Application Layer Gateway (ALG). An ALG acts as a mediator when accessing commonly used Internet applications outside the protected network. For example, for web access, file transfer and multimedia transfer.

ALGs provide higher security than normal packet filtering since the ALGs are capable of scrutinizing all traffic for a specific protocol and perform checks at the higher levels of the TCP/IP stack.

To use an ALG in an IP Rule we must first create an object called ALG object and then assign it to a service which is then used on an IP Rule.

With IP Policies it's quite different. Here we must create an object called a Profile and then assign it to the IP Policy directly. However, in order to be able to assign the profile to the policy we must make sure that the service used in the policy has a protocol that matches the features of the profile. When a profile is applied to an IP Policy the features are still handled by the ALG in the background.

For example if we want to use Web Content Filtering in an IP Policy we want to create a Web Profile. In order to be able to assign the Web Profile to our policy we have to make sure that we have a Web protocol chosen on our service, which would be either HTTP or HTTPS (or both).

We will go into more details on what profiles and protocols are and how to configure them later in this recipe.

Configuring Web Content Filtering

To configure Web Content Filtering we must first create a Web Profile. To do this we must go to the Profile section under the Policies tab in the WebUI as shown in the screenshot below.

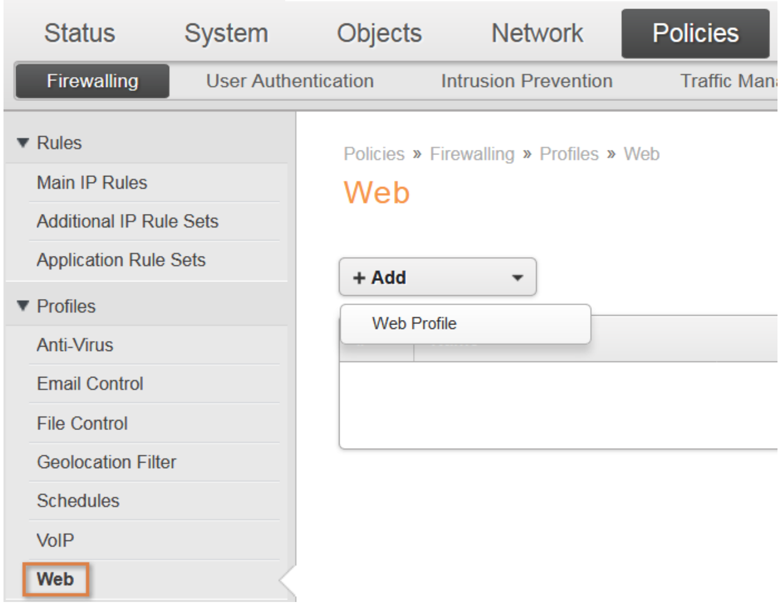
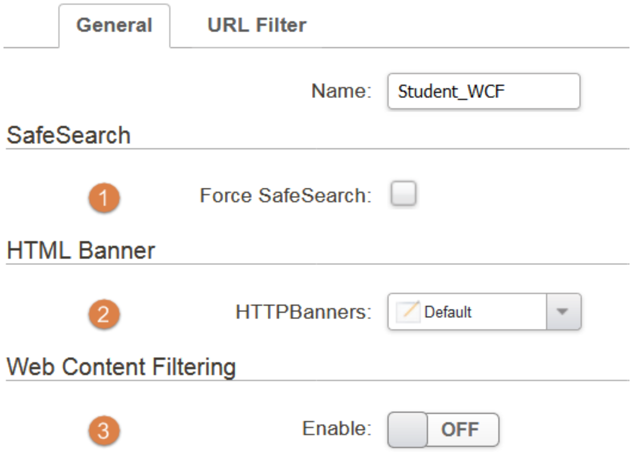


Figure 3.5.2 Location of the Web Profile

Once the Web Profile has been created, we are presented with a few settings. We will go through and describe briefly what each setting does. The settings are shown in the screenshot below.



Name: Student_WCF

SafeSearch

Force SafeSearch: □

HTML Banner

HTTPBanners: Default

Web Content Filtering

Enable: OFF

Figure 3.5.3 Web Profile settings

General settings for the HTTP ALG

The general settings shown above are the following:

1. We have the option to enable SafeSearch to enforce that all client web searches performed with the Google™, Microsoft Bing™ or Yahoo™ search engines are performed using the SafeSearch feature in Strict mode. SafeSearch acts as an automated filter of adult content and potentially offensive content so we will enable this setting.

2. This controls if we want to use any custom made banner files. A banner file is a set of HTML files for various situations that can occur if/when WCF blocks/denies/encounters problems and a specific page is displayed to the user.
   Some users want to create their own error/reject pages and that can be done by creating custom banner files and used on the ALG. We will use the default banner files in our setup.

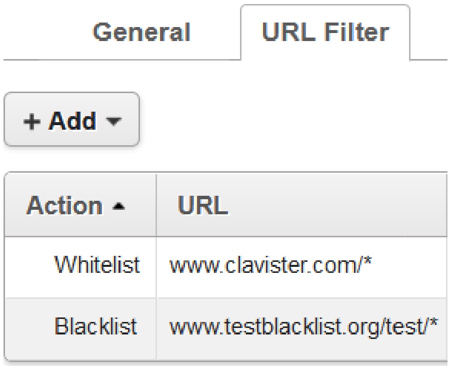
3. This lets us choose if we want to use Web Content Filtering or not. Once WCF is enabled we will get a bunch of extra options for configuring Web Content Filtering that we will go through later in the recipe.

Note

Due to the fact that SafeSearch only works on HTTP and the majority of today's search engines run on HTTPS, this feature has been removed in cOS Core versions 12.00.00 and up.

URL Filter

This tab contains options to whitelist/blacklist specific URLs. It can be very useful to override WCF for specific pages due to one reason or the other. An example of how two URL filters can be configured is shown in the next screenshot.



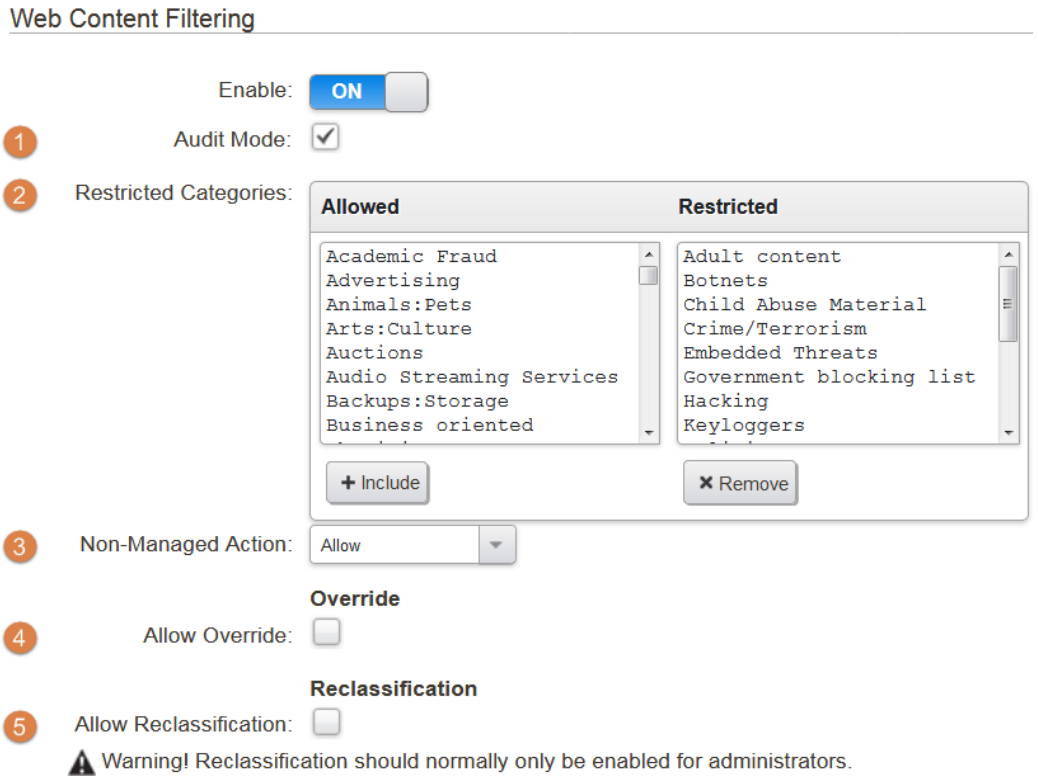
Action ▲	URL
Whitelist	www.clavister.com/*
Blacklist	www.testblacklist.org/test/*

Figure 3.5.4 Using the URL filter

Sometimes, it may be desirable to allow a page in a blocked category. We can use the URL filter's whitelist to override the block. The same is true if you want to override something specific that should be blacklisted but maybe not the entire domain.

Web Content Filtering settings explained

Now it's time to configure the Web Content Filtering. If we click enable on Web Content Filtering we will be given the settings for configuring it. We will go through some of the settings, which are shown in the screenshot below.



Web Content Filtering

Enable: ON

1 Audit Mode: ☑

2 Restricted Categories:

Allowed	Restricted
Academic Fraud	Adult content
Advertising	Botnets
Animals:Pets	Child Abuse Material
Arts:Culture	Crime/Terrorism
Auctions	Embedded Threats
Audio Streaming Services	Government blocking list
Backups:Storage	Hacking
Business oriented	Keyloggers

3 Non-Managed Action: Allow

Override

4 Allow Override: □

Reclassification

5 Allow Reclassification: □

⚠ Warning! Reclassification should normally only be enabled for administrators.

Figure 3.5.5 The Web Content Filtering Settings

Here, we configure how we want to use WCF, what categories we want to allow and deny, and more.

The settings are as follows:

1. This controls if WCF should be run in Audit mode. Since we have not enabled WCF before in our installation, we place it in Audit mode. The reason for this is that each installation's traffic pattern is unique. By first placing WCF in audit mode, the administrator can see what users with what kind of usage will be blocked when we disable audit mode. How long to stay in audit mode will be up to the administrator, but a day or two will most likely be sufficient.

2. This controls what categories we should allow and which categories should be blocked.

3. This is an important setting that controls how cOS Core should handle a situation when a webpage does not have a classification. Should it be allowed or denied? In case of problems, there is always the option to whitelist webpages when the administrator does not want WCF to deny access to them.

4. This determines if we should allow users to override restrictions when going to a blocked category. This means the user gets the option to go to the restricted webpage despite it being initially blocked, but the connection attempt will be logged. It is not recommended to enable this option unless it is for testing purposes.

5. This determines whether you should allow a page reclassification. This means that if a page has the classification News, you get the option to classify it as something else, such as Search Sites. This should rarely be allowed as it does primary two things:

   - It creates a temporary override for the cOS Core WCF URL cache for this page's classification to use the one chosen.

   - It sends a notification to the WCF Databases that this page may require a reclassification.

   Only the administrator should be allowed to reclassify pages.

Note

Some may notice that advertising is not in the blocked category. The reason for that is that many webpages rely on advertisement for their income and many webpages are configured in such a way that if we block the advertisement it will cause the webpage in question to stop loading until the advertisement has been loaded. It will be up to the administrator to decide what should be allowed and blocked. The logs will contain information about what is allowed and blocked so any problems due to this can be quickly investigated.

Advanced: Details about WCF Processing

In order to explain further how WCF works, we will use the diagram shown below in *Figure 3.5.6*.

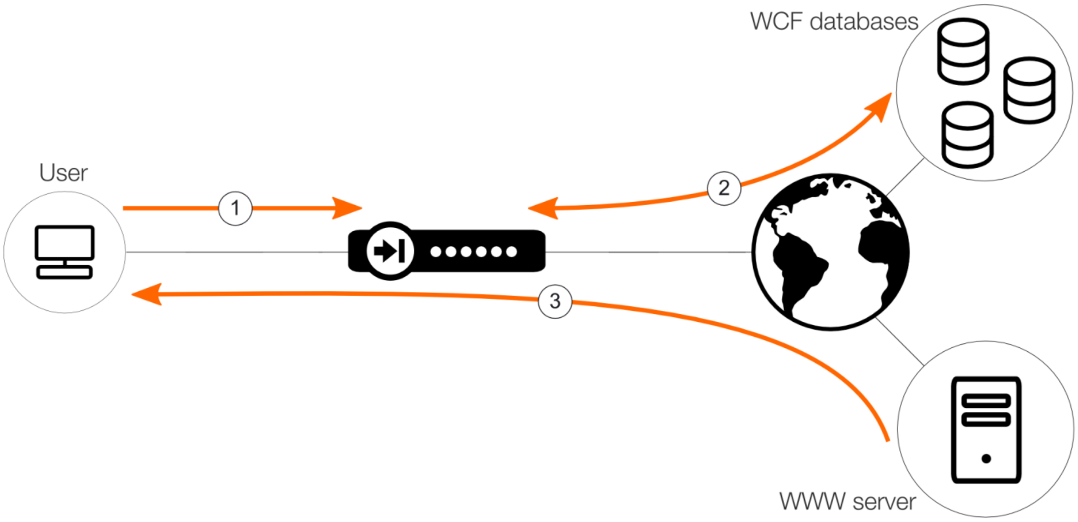


Figure 3.5.6 WCF processing

The processing flow is as follows:

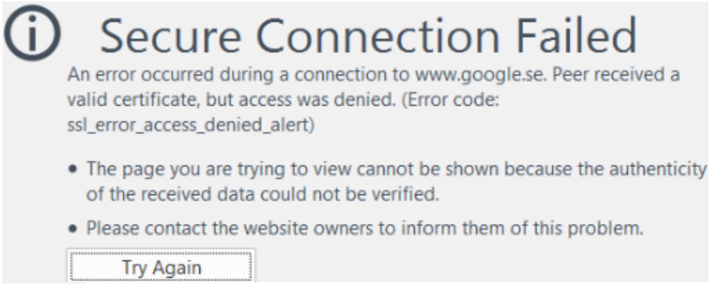
- When a user (**1**) requests access to a web site, cOS Core queries the Dynamic WCF databases (**2**) in order to retrieve the category classification of the requested site.
- Access to the URL can then be allowed or denied based on the currently configured filtering policy.
- If access is allowed, communication between the user and target web server will be established (**3**).
- If access is denied, a web page will be presented to the user explaining that the requested site has been blocked.

Different behavior denying webpages with HTTPS

When using HTTPS it is not possible to insert a webpage to the user showing that the webpage is unreachable when trying to access a restricted category classification.

This is because the traffic is encrypted. What cOS Core can do is to disrupt the data stream and close the active connection.

This will have the effect that the user either gets some sort of page error or that the page stops. An example of such a message is shown below.



Secure Connection Failed

An error occurred during a connection to www.google.se. Peer received a valid certificate, but access was denied. (Error code: ssl_error_access_denied_alert)

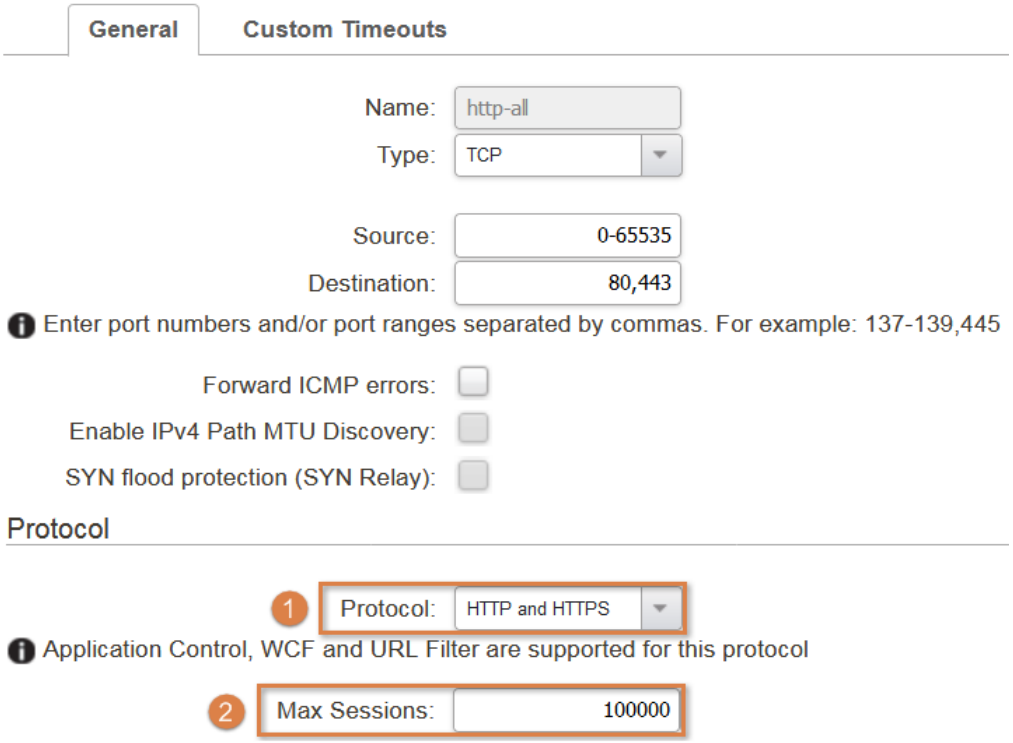
- The page you are trying to view cannot be shown because the authenticity of the received data could not be verified.
- Please contact the website owners to inform them of this problem.

Figure 3.5.7 A typical HTTPS browser loading failure message

This means that users may contact the administrator complaining that a page is not working when it is, in fact, a blocked category.

Continuing WCF configuration

Now that we have finished configuring our Web Profile in the previous steps, we need to assign it to a policy. As mentioned earlier in this recipe we will need to have a service with a web protocol defined to be able to assign our Web profile to our policy. We will keep using the predefined service object "http-all" as it is already configured with the HTTP and HTTPS protocols as shown in the screenshot below.



Name: http-all

Type: TCP

Source: 0-65535

Destination: 80,443

Enter port numbers and/or port ranges separated by commas. For example: 137-139,445

Forward ICMP errors: □

Enable IPv4 Path MTU Discovery: □

SYN flood protection (SYN Relay): □

Protocol: HTTP and HTTPS

Application Control, WCF and URL Filter are supported for this protocol

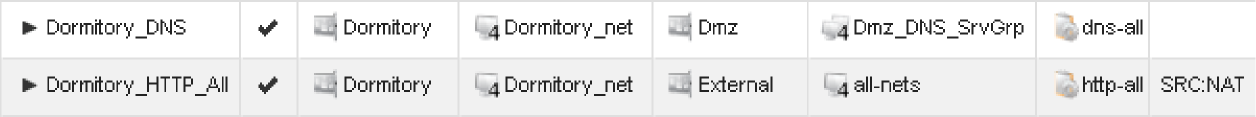
Max Sessions: 100000

Figure 3.5.8 Using the correct Protocol

As seen in the image the http-all object is configured with HTTP and HTTPS as protocol which will be necessary for us in order to assign our web profile (**1**).

Since this is a university, we can expect quite a lot of sessions being used by our users. Configuring the appropriate value for Max Sessions can be tricky, but **a good rule of thumb is to assume 20 sessions per user**. Therefore, if we have 5,000 users, the amount of sessions on the service should be set to 100,000 (**2**).

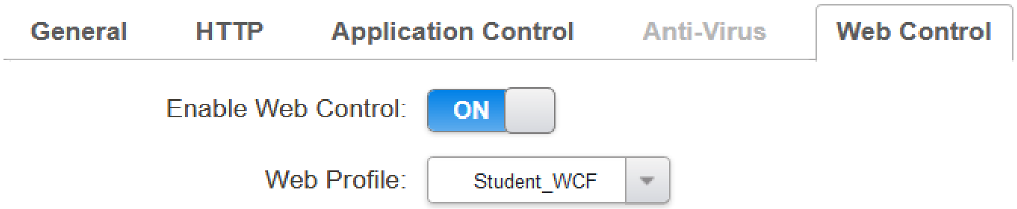
The last step is to use our newly created web profile with an IP policy. Currently our IP policies are as shown in the screenshot below.



▶ Dormitory_DNS	✔	Dormitory	Dormitory_net	Dmz	Dmz_DNS_SrvGrp	dns-all	
▶ Dormitory_HTTP_All	✔	Dormitory	Dormitory_net	External	all-nets	http-all	SRC:NAT

Figure 3.5.9 Configured IP Policies for the Dormitory

We already have a policy to allow HTTP and HTTPS so we simply modify that policy to first enable Web Control under the **Web Control** tab and then assign our Web Profile as shown below.



Enable Web Control: ON

Web Profile: Student_WCF

Figure 3.5.10 Assigning a Web Profile to an IP Policy

Note

If we would use a service that does not have HTTP or HTTPS as protocol, the Web Control tab would not be visible since Core then things that none of the web protocols will be used on this policy. This is why it's necessary for us to use HTTP and/or HTTPS as protocol on our service in order to be able to use our Web Profile.

About the WCF databases

cOS Core Dynamic WCF allows web page blocking to be automated so it is not necessary to manually specify beforehand which URLs to block or to allow.

Instead, cOS Core accesses an external set of databases containing huge numbers of web site URL addresses which are already classified into categories such as shopping, news, sport, adult-oriented and so on.

These external databases are constantly updated with categorized URLs while at the same time, older, invalid URLs are dropped. The scope of the URLs in the databases is global, covering websites in many different languages and hosted on servers located in many different countries.

WCF caches URLs

To make the WCF lookup process as fast as possible, cOS Core maintains a local cache in memory of recently accessed URLs. Caching can be highly efficient since a given user community, such as university students, often surf to a group of similar websites.

Recipe 3.6. Granting different web access privileges based on interface

Objectives

The objective of this recipe is to describe how we can give users different access privileges depending on which interface the user is behind.

We will create multiple Web Profiles that are configured differently from each other, giving access or restrictions to specific categories using Web Content Filtering (WCF).

Detailed Discussion

Creating additional Web Profiles

After implementing WCF as described in *Recipe 3.5. Restricting web access with Web Content Filtering*, there are restrictions on some of the undesirable web categories for our students. We now want to implement some restrictions for our teachers and Wi-Fi users as well.

We want to implement additional restrictions by adding more blocked categories. For example, blocking remote desktop, chatrooms, and so on.

In the end, it will be up to the administrator and the university to determine the policy on what should be allowed or not.

If we look at our three primary university example groups which is Student/Dormitory, Teachers and Wi-Fi, it would be prudent to give them access to different categories.

A teacher, for instance, should be given less restrictions than a student as there may be school work, projects or other things that may require the teacher to have additional access to these resources without having to go to through the network administrator every time. We will discuss special override access later.

Table 3.6.1 below, shows an example of how we can configure different categories for various interfaces and groups.

Interface	Options	Blocked Category
TEACHERS	Override	Adult Government blocking list Malicious Violence/undesirable
STUDENTS/DORMITORY	SafeSearch	Adult Crime/Terrorism Drugs/alcohol Gambling Government blocking list Malicious Remote control/ desktop Swimsuit/lingerie Violence/undesirable
WI-FI	SafeSearch	Adult Chatroooms Crime/Terrorism Drugs/alcohol Gambling Government blocking list Malicious Music download Remote control/ desktop Swimsuit/lingerie Violence/undesirable

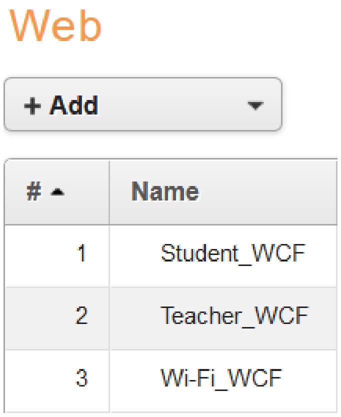
Table 3.6.1 Blocked categories for Different Groups

In order to implement our customized access, we need to create additional Profiles for use in our IP rule set.

Remember the 3 steps needed to create and use a Profile in our IP policies:

1. Create a Profile object.
2. Associate a Service object that is configured with the protocol to use in the IP Policy.
3. Associate the Profile with an IP Policy.

We proceed to create our custom Web Profiles as shown in the next screenshot.



# ▲	Name
1	Student_WCF
2	Teacher_WCF
3	Wi-Fi_WCF

Figure 3.6.2 Summary of Web Profiles

DMZ and ADMIN access rights

Some readers might notice that we have not added any Web Profiles for DMZ, Administrators or the Lab.

The DMZ contains the servers for incoming connections to the university website, domain controllers, storage devices, backup systems, mail servers and so on. They are primarily controlled by the administrator with very limited external access. The need for the ALG behind this interface is limited, but at the end it will be up to the administrator to decide what services to use and what kind of access level that should be allowed.

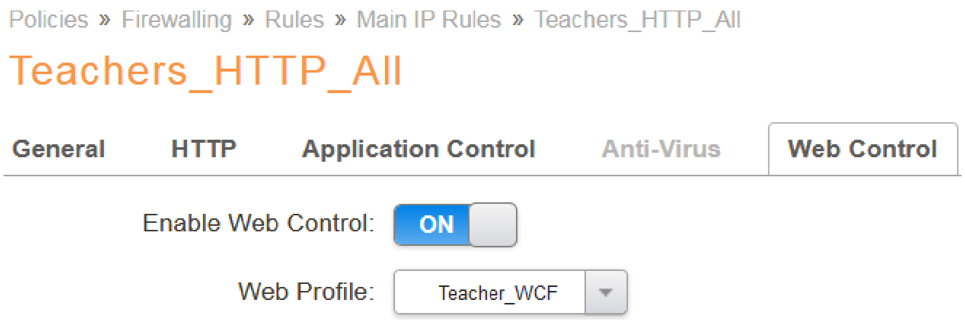
The administrators currently have unrestricted access to the Internet without any restrictions for either category or port/protocol. But are administrators infallible? No, they are not. Even though they administer the network, there is no direct reason NOT to restrict them as well. If the need arises for the administrators to get access to additional ports/protocols they can easily modify the cOS Core IP rule set to accommodate this need.

This is one reason why hackers target administrator accounts, because of their wide network access privileges.

For now we have changed the external access for our university administrator to only allow DNS and HTTP/HTTPS without an ALG. They still have full access to every internal resource, network and interface.

Using individual Web Profiles with IP Policies

All we have left to do now, is to use our newly created Web Profile objects on our corresponding IP policies. Doing this is shown in the screenshot below.


Policies » Firewalling » Rules » Main IP Rules » Teachers_HTTP_All
General
HTTP
Application Control
Anti-Virus
Web Control

Teachers_HTTP_All

Enable Web Control: ON

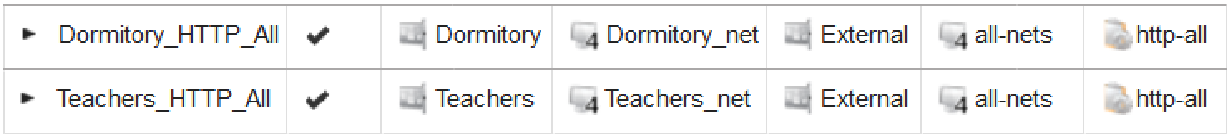
Web Profile: Teacher_WCF

Figure 3.6.3 Using Web Profiles

Note

Remember that to be able to assign a Web Profile to an IP Policy, our service object must have HTTP and/or HTTPS defined as protocol. We use the default object http-all in this example since it has HTTP and HTTPS defined as protocol.

We must create IP policies for each interface and network combination and then use the correct Web Profile to give each interface and network the correct Web Content Filtering access level. For example, the 'Student_WCF' Web Profile must be used on the network that our students are connected to, which in the screenshot example below is Dormitory and Dormitory_net.



► Dormitory_HTTP_All	✓	Dormitory	Dormitory_net	External	all-nets	http-all
► Teachers_HTTP_All	✓	Teachers	Teachers_net	External	all-nets	http-all

Figure 3.6.4 Policy examples for Web Profiles

When this is done for all involved interfaces and networks, we have now effectively assigned different access rights to the Students, teachers and Wi-Fi users, depending on which network they are connected to.

We will continue adding more security and functionality to our university network and this is a point that we will discuss in other recipes as we progress through the book. The lab network is a different matter, however, and we will go into more details regarding the lab network later.

Recipe 3.7. Configuring Anti-Virus protection

Objectives

The objective of this recipe is to further strengthen the security of the university network. For this recipe we will discuss how to activate Anti-Virus scanning using Anti-Virus Profiles.

The cOS Core Anti-Virus functionality protects against any malicious code carried in files being downloaded to clients via a Clavister Next Generation Firewall. The following can be scanned for viruses:

- Any files downloaded via the Clavister firewall. For example, files downloaded using HTTP transfer or FTP or as an attachment to email delivered via SMTP.
- Scripts contained within webpages delivered via HTTP.
- URLs contained within webpages delivered via HTTP.

Anti-virus scanning functionality can be enabled for file downloads associated with the following Protocols:

- HTTP
- FTP
- POP3
- SMTP

Note

Anti-Virus can be activated on any Policy using a service object that is configured with HTTP, SMTP, POP3 or FTP as protocol but in this example we will only focus on Anti-Virus in HTTP. The general options and functionality are similar for all protocols that support Anti-Virus.

Detailed Discussion

How to activate Anti-Virus scanning on an IP Policy

To activate Anti-Virus on an IP Policy, we must first create an Anti-Virus profile as in the WebUI screenshot below.

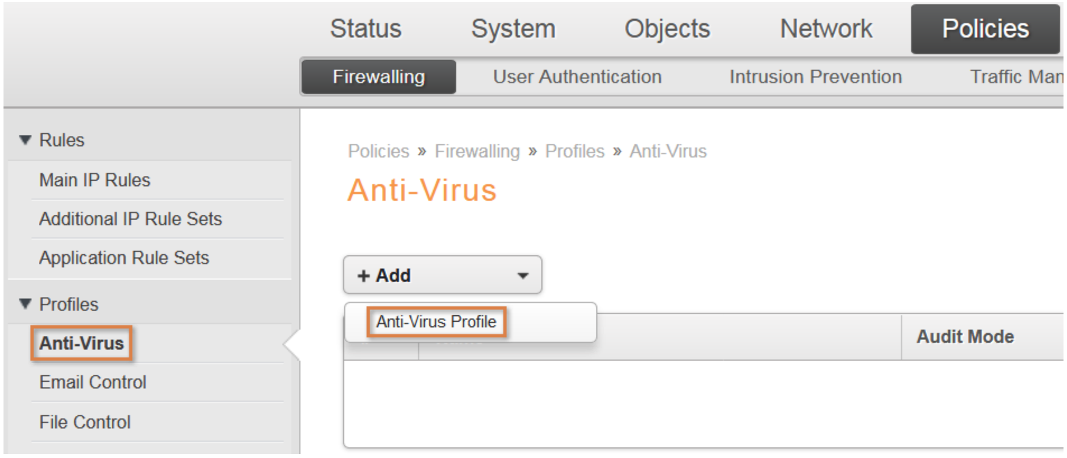
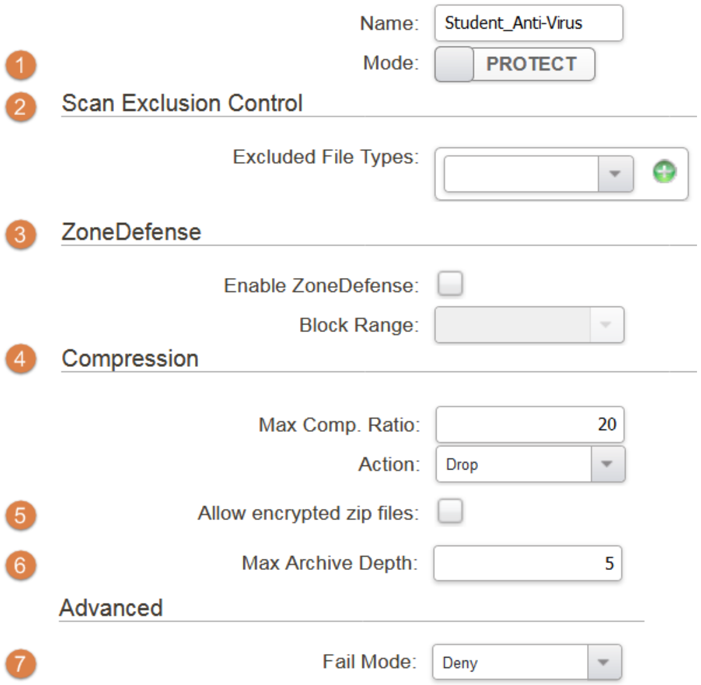


Figure 3.7.1 Creating an Anti-Virus Profile in the WebUI

The Anti-Virus Profile has many different options for us to configure. We will go through each option shown in figure 3.7.2.



Name: Student_Anti-Virus

Mode: PROTECT

Scan Exclusion Control

Excluded File Types:

ZoneDefense

Enable ZoneDefense: □

Block Range:

Compression

Max Comp. Ratio: 20

Action: Drop

Allow encrypted zip files: □

Max Archive Depth: 5

Advanced

Fail Mode: Deny

Figure 3.7.2 Creating an Anti-Virus Profile in the WebUI

When activating Anti-Virus, we first have the option to select in which mode (**1**) it should be run. We have two options, Audit and Protect. **Audit** means that Anti-Virus will only scan the traffic. if it detects a virus or other threat, it will generate a log event but will not interfere with the traffic. This can be very useful if we are only interested in evaluating the status of a network without doing any active handling of erroneous behavior.

In **Protect** mode we will actively stop the file from being downloaded. The AV engine is stream based, meaning it will continuously scan the file as it is being downloaded, if a virus or other malicious content has been detected it will inject a message into the data stream and drop the connection.

Scan Exclusion Control (**2**) means the ability to select whether a specific file type should be excluded from the Anti-Virus scanning. Excluding files from scanning should be done with caution.

ZoneDefense (**3**)is a feature for isolating virus infected hosts and servers on a local network. While the Anti-Virus takes care of blocking inbound infected files from reaching the local network, ZoneDefense can be used for stopping viruses to spread from an already infected local host to other local hosts. When the AV engine has detected a virus, the firewall will upload blocking instructions to the local switches and instruct them to block all traffic from the infected host or server.

When scanning compressed files (**4**), cOS Core 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 cOS Core resources and noticeably slowdown throughput.

To prevent this situation, the administrator should specify a **Compression Ratio Limit**. If the limit of this ratio 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 the following:

- **Allow** - The file is allowed through without virus scanning.
- **Drop** - Drop the file.

The **Allow encrypted zip files** option (**5**) is what to do when we encounter files which are password protected or encrypted. The default behavior is that we should not allow such files as we cannot scan them. It is however not that uncommon to password protect, for example, zip files. So it will be up to the administrator to determine whenever he should allow or deny such files to pass through the firewall.

In our university scenario example, we will allow password protected files to be sent on the teacher's network but denied on every other network.

cOS Core can perform virus scanning on compressed files within other compressed files. The level of nesting which is allowed is controlled by the **Maximum archive depth** setting (**6**). For example, if this setting is set to zero then any compressed file will always cause a fail condition. If set to a value of one, compressed files will be scanned but any compressed files containing other compressed files will fail to be scanned. A value of two allows a single nesting level of compressed files within compressed files, with both levels being scanned and so forth.

Our last setting is the **Fail Mode** setting (**7**). This setting lets us control whether we want to drop or allow a file if the anti-virus scan for any reason would fail. If this option is set to Allow then an event such as the virus database not being available or the current license subscription expiring 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 occurred.

Important: A file can appear downloaded even if a virus is found

An important aspect to keep in mind when it comes to Anti-Virus is that even though we can abort a file download by the user, we cannot always direct the user to a web page saying "virus found". However, what we can do is to deliberately corrupt the file by aborting the download and inject the message "Virus found" at the end of the file once a virus has been detected.

Therefore, if a user complains that a file download is not working, it may be that a virus has been found in this particular file. Either check the logs generated by cOS Core related to Anti-Virus or try open the corrupted file in a hex-edit or similar and look at the end of the file to check if it contains a note saying "virus found".

So don't exclude the file type just because a user is having a problem with one file, it may be that something was deliberately done to the file.

Below is an example of what might be inserted into an HTML file by cOS Core:

```
<html>
       <head>
               <title>Virus Found</title>
       </head>
       <body>
               <h1>A virus has been detected</h1>
               <b>The rest of this file will be blocked</b>
               Infected File: eicar_com.zip
```

If the virus was detected on a webpage, the user would be presented with a simple page containing the above message. However, if it is a file being downloaded, that is not possible and instead the file will be deliberately corrupted by cOS Core.

cOS Core Anti-Virus should not replace other Anti-Virus

It is very important to be aware that cOS Core Anti-Virus scanning should never completely replace local Anti-Virus software installed on a client PC. cOS Core will not perform the same level of detailed file scanning and investigation that a locally installed Anti-Virus program can perform because cOS Core can only examine the network communication.

cOS Core Anti-Virus scanning should be the first line of defense, but not the last.

A note about Anti-Virus scanning and HTTPS

With HTTPS, the traffic is encrypted and Anti-Virus will not be able to scan the data. Therefore the Anti-Virus can not be activated on a policy when using HTTPS as protocol (this also applies for the "HTTP and HTTPS" protocol).

Anti-Virus policy for the university

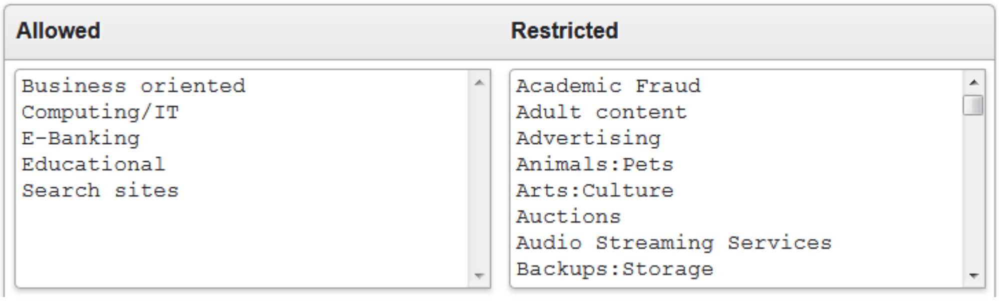
In *Recipe 3.5. Restricting web access with Web Content Filtering* and *Recipe 3.6. Granting different web access privileges based on interface*, we mentioned that we did not activate the Web Profile for administrators or the DMZ in order to give them as much unrestricted access as possible.

However, is there really any good reason not to activate at least Anti-Virus for the admins and DMZ servers? Let us continue adding additional access levels and protection for all our activate interfaces, including ADMIN and DMZ.

For the administrators, the only feature we activate is Anti-Virus for HTTP. We also allow the forwarding of encrypted files through the Anti-Virus.

There will always be situations where this may need to be overridden due to one reason or the other. We will go into more details about overriding access levels and user authentication in a later chapter.

For the DMZ, we implement both a web profile and an anti-virus profile that is customized for the DMZ. On the web profile we will only activate Web Content Filtering and we will only allow a very limited number of categories, as shown below.

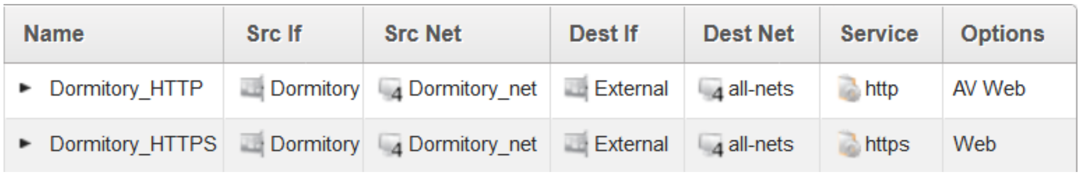


Allowed	Restricted
Business oriented	Academic Fraud
Computing/IT	Adult content
E-Banking	Advertising
Educational	Animals:Pets
Search sites	Arts:Culture
	Auctions
	Audio Streaming Services
	Backups:Storage

Figure 3.7.3 WCF categories for the DMZ network

On the anti-virus profile we do not allow encrypted files to pass through, so this particular option is disabled.

Since the anti-virus can not be applied on a policy that has HTTPS as protocol we will have to split our HTTP-ALL policies into two separate policies. One for HTTP where we apply our web profile as well as anti-virus profile. And a second policy for HTTPS where we will only apply our web profile as you can see under the "Options" column in the screenshot below.



Name	Src If	Src Net	Dest If	Dest Net	Service	Options
► Dormitory_HTTP	Dormitory	Dormitory_net	External	all-nets	http	AV Web
► Dormitory_HTTPS	Dormitory	Dormitory_net	External	all-nets	https	Web

Figure 3.7.4 Separating the HTTP/HTTPS policies

Recipe 3.8. Network hardening when using FTP

Objectives

The objective of this recipe is to provide details on how to configure the File Transfer Protocol (FTP) custom settings and how the they can be used to significantly lower the number of ports that need to be open to/from a server, either in the local network or out on the Internet.

Detailed Discussion

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.

FTP operates in two modes: **Active** and **Passive**. These determine the role of the server when opening data channels between client and server.

- **Active Mode**

  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.

- **Passive Mode**

  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.

FTP security concerns

Both active and passive modes of FTP operation present difficulties for cOS Core. Consider a scenario where an FTP client on the internal network connects through the Clavister firewall to an FTP server on the Internet. The IP policy is then configured to allow network traffic from the FTP client to port 21 on the FTP server.

When **Active Mode** is used, cOS Core does not 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 there are no incoming policies configured that will allow this connection.

The port number used for the data channel is dynamic so the only way to solve this is to allow traffic from all ports on the FTP server to all ports on the FTP client. Opening more than 64,000 incoming ports is not an alternative (clients usually avoid port 0 to 1023 due to those being system ports).

When **Passive Mode** is used, the firewall does not need to allow connections from the FTP server. On the other hand, cOS Core still does not know what port the FTP client will try 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 security problem that we may need to open as many as 64,000-plus ports in the outgoing direction.

The cOS Core FTP Solution

When we apply the FTP protocol as service for the IP policy controlling FTP traffic, customizable settings regarding FTP will become available on this Policy that can solve this problem by examining both the command and data channel. When these settings are enabled cOS Core will run something called an FTP Application Layer Protocol in the background also known as FTP ALG. The FTP ALG will know what port(s) the server and client will use and will open the required ports between the client and server (or the other way around) without the need to open up 64,000 ports in either incoming or outgoing direction.

The FTP ALG examines both the command and data channel, it will immediately drop the traffic if the command or data sent through the ALG is not determined to be FTP. The ALG will effectively make sure that any incoming/outgoing connection through the ALG is valid.

Advantages of using the FTP ALG Hybrid Mode

An important feature of the cOS Core FTP ALG is its automatic ability to perform on-the-fly conversion between active and passive mode so that FTP connection modes can be combined. Passive mode can be used on one side of the firewall while Active mode can be used on the other. This type of FTP ALG usage is sometimes referred to as *hybrid mode*. The advantage of hybrid mode can be summarized 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 recommended mode for servers).
- When an FTP session is established, the Clavister Next Generation Firewall will automatically and transparently receive the passive data channel from the FTP client and the active data channel from the server, and correctly tie them together. This is illustrated next in *Figure 3.8.1*.

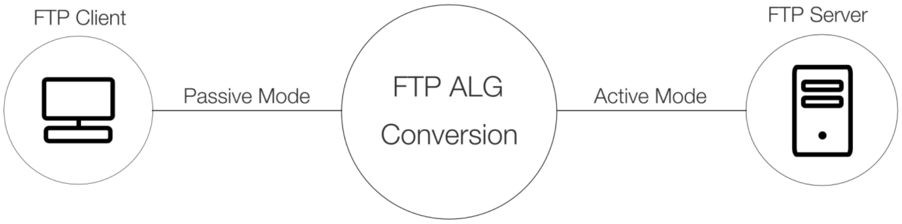


Figure 3.8.1 FTP ALG hybrid mode conversion

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.

The FTP ALG cannot be used with encrypted FTP traffic

It is important to remember that the FTP ALG will not support encrypted FTP traffic, such as SFTP or FTPS.

Without the FTP ALG, allowing encrypted FTP traffic means potentially opening up a large number of incoming ports. If the need arises for encrypted FTP, it is recommended to configure the server and client to only use a specific port range in order to try to limit the amount of ports opened.

The FTP settings and the different mode variations

First we will look at the Data Channel Restriction settings found under the FTP tab. We will need to use these settings allow or deny different modes for the client and server depending on how the FTP client and FTP server are configured and what mode they are using. A summary of the different combinations of settings that work, can be found below in Table 3.8.2

#	Client Setting	Allowed Client Modes	Allowed Server Modes	Server Mode
1	Active or Passive	Active and Passive	Active	Active
2	Active or Passive	Active and Passive	Active and Passive	Active or Passive
3	Passive	Passive	Active and Passive	Active or Passive
4	Passive	Passive	Active	Active

Table 3.8.2 Allowed modes based on server and client configuration.

The above table may be difficult to understand so let's add a few examples when the different settings are used. One very important aspect to be aware of when dealing with the FTP ALG is that it creates two connections. First it will be a connection from the FTP client to the ALG, then from the ALG to the FTP server, so it behaves similar to a proxy.

Example 1: Secure a FTP-server located on our DMZ

- We want the server to run in active mode since it will then be the initiator of the data channel. It is more secure to open an outbound connection then to listen on many incoming ports.

- This means that we want to suggest active mode towards the server (**#1** in the table above).

- Now, if we were to allow only active mode to the clients they would need to open high ports and if NAT is used we would need port forwarding on the client because the server side would initiate connection.

- To solve both the above problems, we must allow clients to use passive mode so they will be the initiator of the data-channel. As the combination seen in **#1**.

- If the client prefers passive mode and our server uses the active mode, we will run into a situation where the modes not match. The ALG will sort this out by using hybrid mode to tie the data channels together (as mentioned previously).

- Using passive mode on the client still requires opening high ports in cOS Core. The ALG will control which port(s) to use through the command-channel, only allowing traffic to that port from the specific client. This is the primary strength of the FTP ALG since there is no need to open up all ports between 1,024 and 65,535 to make FTP work.

Example 2: FTP Client user decides what mode to be sent to the FTP server

- We want to accept both passive and active modes on the client and server side. (**#2** in the table above)
- The ALG will forward to the server the mode that the client used to connect with.
- Can be useful if the administrator wants to give the FTP client user the option to select which mode he wants to use. No hybrid modes will be used with this combination.

Example 3: Secure users when using FTP clients to connect to servers on the Internet

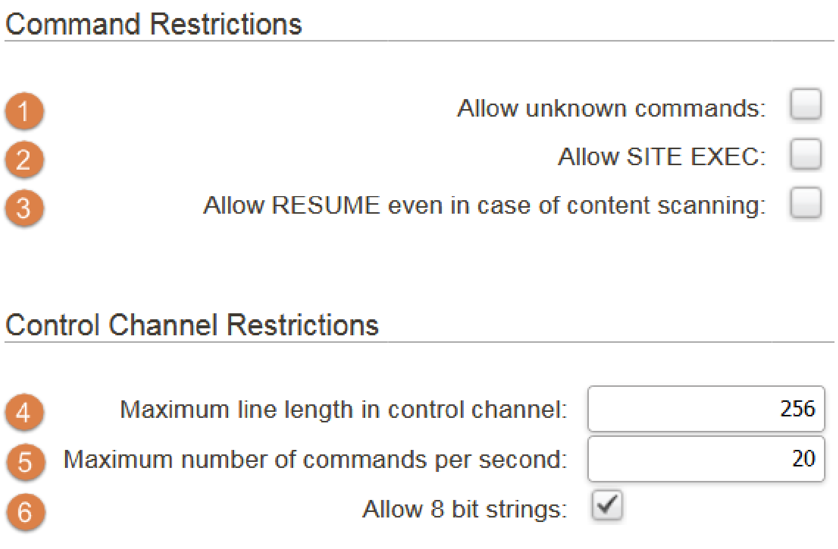
- We want the clients to run in passive mode as we want client to initiate the data-channel. We don't want to open listening ports on the firewall or the client.
- This means the ALG should only accept passive mode on the client side (**#3**). If the client attempts to use active mode then the connection will not be able to be completed since active mode for client has not been allowed.
- The ALG will offer passive mode to an outside server but if the server rejects this request, the ALG will instead offer active mode and then run in hybrid mode.

Example 4: FTP-server and Client are both behind the firewall

- The client must use passive mode and server must use active mode (**#4**).
- As both servers and clients are under our control and we have configured them in this particular way, this narrow setup will work.

Additional FTP options

There are some additional settings under the FTP tab that the administrator may want to either change or at least be aware of, as shown in the next screenshot.



Command Restrictions

1 Allow unknown commands: □

2 Allow SITE EXEC: □

3 Allow RESUME even in case of content scanning: □

Control Channel Restrictions

4 Maximum line length in control channel: 256

5 Maximum number of commands per second: 20

6 Allow 8 bit strings: ☑

Figure 3.8.3 FTP command and control channel options

Allow unknown commands (**1**) - This can allow FTP commands that are not part of the standard FTP command set (defined in RFC 959).

Allow SITE EXEC (**2**) - Some old versions of FTP servers support this command, which permits access to the server itself. It is recommended to have this option turned off so that the ALG blocks it.

Allow RESUME (**3**) - The resume command can be allowed to execute even if content scanning terminated the connection. For example, it might have been blocked because we do not allow .exe files to be transferred using FTP. It is recommended to leave this option disabled.

Maximum line length in control channel (**4**) - This restricts the maximum command line a client can send to the server. Creating very large control channel commands can be used as a form of attack against a server by causing a buffer overrun. This restriction combats this threat. The default value is 256.

If very long file or directory names on the server are used then this limit may need to be raised. The lower the limit, the better the security.

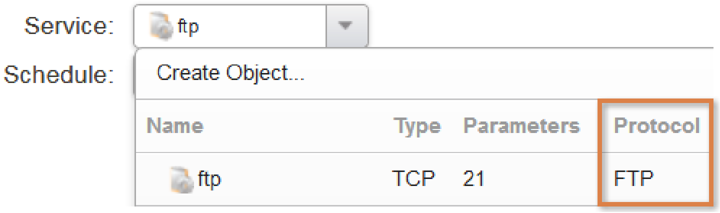
The **Commands per second** (**5**) option can be used to prevent automated attacks against the FTP server. The default is 20 commands per second.

The option **Allow 8-bit strings in the control channel** (**6**) determines if 8-bit characters are allowed in the control channel. Allowing 8- bit characters enables support for filenames containing international characters. For example, accented or umlauted characters, such as *åäö*.

Creating IP policies for outgoing traffic

Unlike the HTTP/HTTPS configuration discussed previously, FTP does not use profiles. Instead we will configure the FTP settings directly on the policy

To enable the FTP settings in a policy we need to use a service that has the Protocol FTP assigned. We will use the already existing service "ftp" as it is preconfigured with FTP as Protocol.



Service: ftp

Schedule:

Create Object...

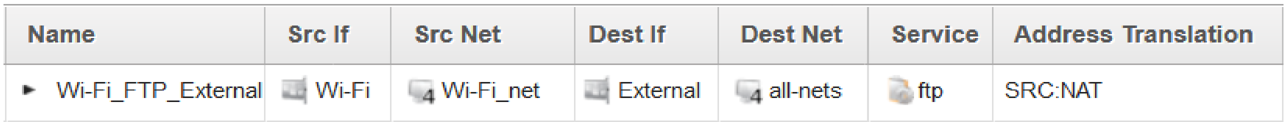
Name	Type	Parameters	Protocol
ftp	TCP	21	FTP

Figure 3.8.4 Using a service with FTP Protocol

Note

By "outgoing traffic", we mean traffic initiated from any internal network out towards the Internet.

We create one outbound IP policy for all our internal clients. We will begin with Wi-Fi as seen in the next screenshot.



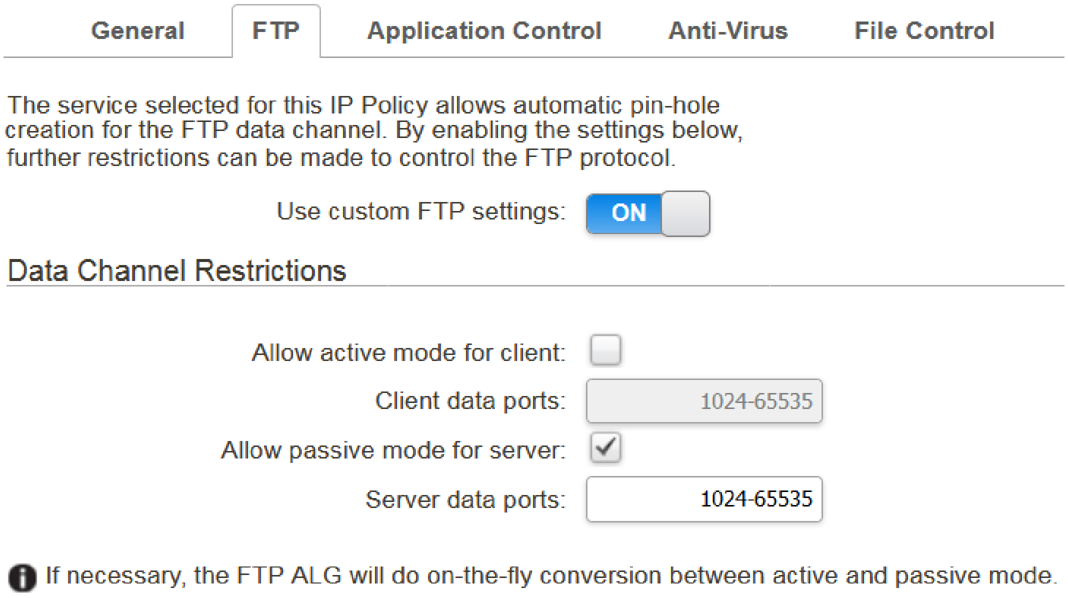
Name	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
► Wi-Fi_FTP_External	Wi-Fi	Wi-Fi_net	External	all-nets	ftp	SRC:NAT

Figure 3.8.5 Outbound FTP policy for the Wi-Fi network

Since we have applied the ftp protocol on this policy we are able to access the custom FTP settings under the FTP tab.

What we want to do is secure our users when they are using FTP clients to connect to servers on the Internet. This matches our example number 3 that was mentioned earlier, where we mentioned that we will want to allow only Passive mode for the clients and both Passive and

Active mode for the server. To set this in our IP policy we go to the FTP tab of the IP policy and enable the use of custom FTP settings. Then we will un-check the box for Allow active mode for client and check the box for Allow passive mode for server as seen in the next screenshot.



The service selected for this IP Policy allows automatic pin-hole creation for the FTP data channel. By enabling the settings below, further restrictions can be made to control the FTP protocol.

Use custom FTP settings: ON

Data Channel Restrictions

Allow active mode for client: □

Client data ports: 1024-65535

Allow passive mode for server: ☑

Server data ports: 1024-65535

If necessary, the FTP ALG will do on-the-fly conversion between active and passive mode.

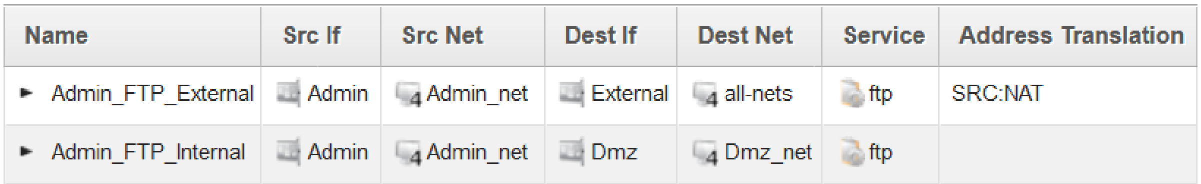
Figure 3.8.6 FTP options for outgoing traffic

Repeat this process for our Teachers, Dormitory and Admin networks so that we cover all of our internal clients.

Note

Passive mode is always allowed for clients and Active mode is always allowed for servers.
Allowing passive mode for server therefore means that both active and passive mode will be allowed. The same applies for the client modes.

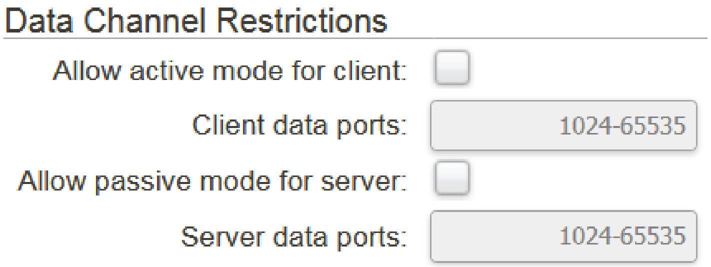
For our administrators, we need one additional FTP policy as they will have a need to access the FTP server(s) by directly connecting to the DMZ. This extra policy is shown in the next screenshot. Since this is towards an internal network, we do not have any need to apply address translation to the source traffic with, for example, using NAT. Therefore we will set address translation to none.



Name	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
► Admin_FTP_External	Admin	Admin_net	External	all-nets	ftp	SRC:NAT
► Admin_FTP_Internal	Admin	Admin_net	Dmz	Dmz_net	ftp	

Figure 3.8.7 Internal policy for the Admin network

In this case both the FTP client and the server will be located behind the firewall so we can use the configuration mentioned in example number 4. We will un-check both the check boxes under Data Channel Restrictions.



Data Channel Restrictions

Allow active mode for client: □

Client data ports: 1024-65535

Allow passive mode for server: □

Server data ports: 1024-65535

Figure 3.8.8 Data channel restrictions for internal traffic.

Creating IP policies for incoming traffic

Now that we have created the outgoing IP policies, we need an IP policy to allow incoming traffic from the Internet to flow to our FTP server in the DMZ. This is illustrated next in *Figure 3.8.9*.

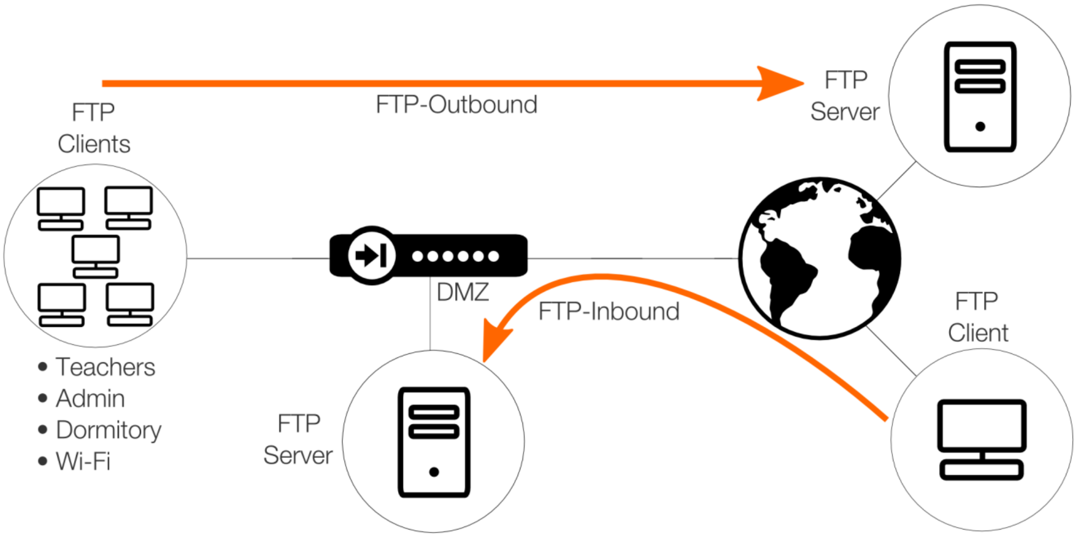
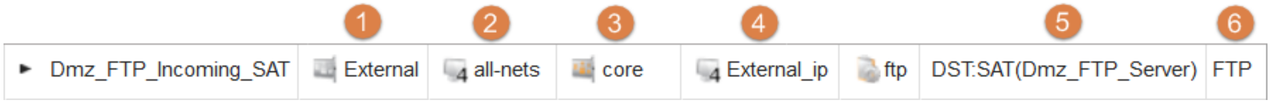


Figure 3.8.9 FTP-Outbound for external and FTP-Inbound for incoming connections

In order to allow access to and secure an FTP-server located on our DMZ we will create a policy with the properties shown below.



▸ Dmz_FTP_Incoming_SAT	External	all-nets	core	External_ip	ftp	DST:SAT(Dmz_FTP_Server)	FTP

Figure 3.8.10 Properties for incoming FTP traffic policy

As we can see, the properties for this policy are quite different from all our previous policies. This is because the order of traffic has been reversed. Instead of initiating traffic from the inside network we now have a situation where we have users on the Internet wanting to access our FTP server in the DMZ.

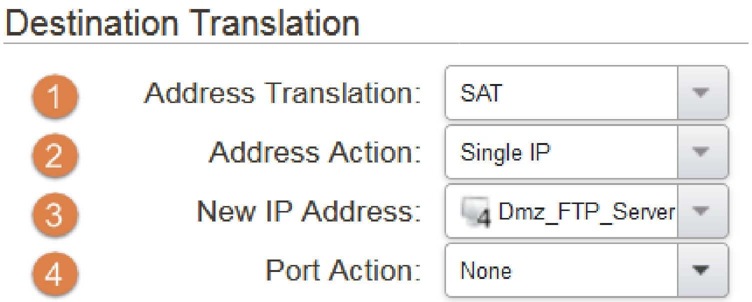
To accomplish this, we need to create a policy that handles traffic arriving from the Internet. Since *EXTERNAL* is the name of our external Internet interface, that will be the source interface (**1**). We do not know the IP of all the clients connecting from the Internet so we have set the source network to *all-nets* (**2**). This can of course be restricted using a single IP, network or group if limiting access is preferred, but in our example we have a public FTP server accessible by everyone so the source network is set as *all-nets*.

The destination interface is Core (**3**) as the destination network (**4**) is the shared IP address used by the External interface and this is the IP address the clients will attempt to connect to. By default, that IP address is Core routed.

Note

The Core interface is described in more detail in Chapter 2: Basic Setup .

Then we have the address translation information (**5**). In this case we have DST (destination) translation, as compared to our previous NAT policies. This is shown below.



Destination Translation

1 Address Translation: SAT

2 Address Action: Single IP

3 New IP Address: Dmz_FTP_Server

4 Port Action: None

Figure 3.8.11 The destination translation for incoming FTP traffic

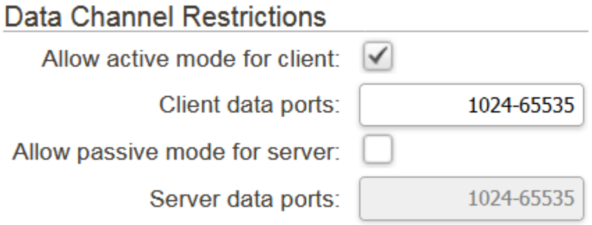
By default, Address translation is set to be none. To use destination translation we need to set this to SAT (**1**). It is possible to use SAT on source translation as well but that is rarely used when NAT can be used instead. There are some scenarios where a source SAT may be useful but that will be discussed in a later chapter.

The Address Action (**2**) can be set to either Single IP or Transposed. The most commonly used one is Single IP and in our scenario we will also use Single IP as we want all FTP traffic directed towards our external IP address to be forwarded to a single IP which is the IP address of our FTP server on the DMZ.

Transposed can be used if you want to transpose an entire network to another network, also known as many-to-many mapping.

The new IP address (**3**) will be the FTP server with a private IP address located in the DMZ. Since we are not changing the destination port (**4**) we will leave this particular option blank. This means that the original port will be used (in this case, FTP port 21).

The last property we need to configure on our IP Policy is the FTP settings. In order to secure our FTP-server located behind our DMZ we will follow example number 1 which means we will allow both active and passive mode for the clients but we will only allow active mode towards the server as seen in the next screenshot.



Data Channel Restrictions

Allow active mode for client: ☑

Client data ports: 1024-65535

Allow passive mode for server: □

Server data ports: 1024-65535

Figure 3.8.12 Data channel restrictions for incoming traffic

Note

We refer to any traffic initiated on the Internet towards cOS Core as incoming traffic.

Users can now connect to the external IP of cOS Core on port 21 in order to access the public university FTP server.

Recipe 3.9. Public FTP server access

Objectives

The objective of this recipe is to describe how to solve a scenario that many administrators may run into.

In the previous recipe, we discussed the use and configuration of FTP clients and servers. This recipe will use the same IP policies as a basis for describing how to solve two problems that can occur.

The policies we currently have created for both FTP and HTTP have the following two major problems:

1. It is not possible to connect to the university FTP server from any other interface than the EXTERNAL interface.

2. The IP policies for outgoing HTTP only trigger when users want to reach Internet addresses past the external interface. If a user wants to connect to the university FTP server (or webpage) it will not work since the destination interface will not be the EXTERNAL interface but rather the Core interface.

Note

This recipe uses the previous recipe as its basis, but the solution can easily be adapted to be used on any other similar features/rule sets such as that described for the recipe describing a web server located in a DMZ in Chapter 2: Basic Setup .

Detailed Discussion

To solve this problem we will create a new set of IP policies that will handle this particular situation. Our currently configured FTP policies are as shown next in *Table 3.9.1*.

Name	Src Iface	Src Net	Dest Iface	Dest Iface	Service	FTP Client modes	FTP Server modes
Dmz_FTP_ Incoming_SAT	External	all-nets	core	External_ip	ftp	Active and Passive	Active
Wifi_FTP_ External	WiFi	WiFi_net	External	all-nets	ftp	Passive	Active and Passive

Table 3.9.1 FTP IP policies for the External and Wi-Fi interfaces

If we attempt to connect from a host in the Wi-Fi network towards "External_ip" our SAT policy will never trigger due to the reasons listed in **1** and **2** at the beginning of this recipe.

Note

Configuring incoming traffic policies using Destination SAT is discussed at the end of Recipe 3.8. Network hardening when using FTP .

An example of problematic packet flow

A user behind the WiFi network wants to access the FTP server located on the public IP of the university, he attempts to connect to the "External_ip".

His source interface will be WiFi, so we can now see that our incoming SAT FTP policy will not trigger as it is only configured to trigger if the source interface is "EXTERNAL".

The outgoing Wi-Fi policy will not trigger either because the destination interface where "External_ip" is routed is Core and not EXTERNAL.

Therefore, the end results would be that the traffic will be dropped as no IP policy matches.

Two possible solutions

This problem has multiple solutions. There is no best way to solve this as it will be up to the administrator to decide how their policy should be implemented.

1. Create a new policy with Destination SAT that specifically triggers for each internal network.

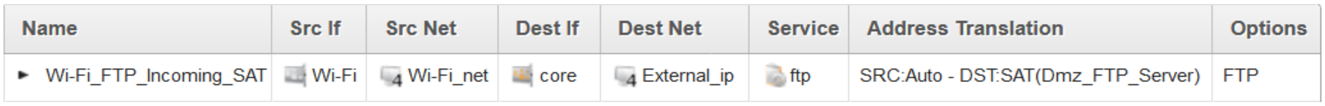
2. Modify existing policies to include the involved source and destination interfaces.

Both of the above solutions have their pros and cons:

Solution 1 has the advantage that the policies will be easier to read and understand. The disadvantage is that it requires more policies to be created to handle the various scenarios and interfaces.

Solution 2 has the advantage that the rule set will be more streamlined as we do not need to add as many IP policies. The disadvantage is that the configuration will be more difficult to read and understand.

We will use **Solution 1** in this chapter. We create a new SAT policy for all internal interfaces and networks that should be able to access the public FTP server. An example of what the policy would look like for the Wi-Fi network.



Name	Src If	Src Net	Dest If	Dest Net	Service	Address Translation	Options
► Wi-Fi_FTP_Incoming_SAT	Wi-Fi	Wi-Fi_net	core	External_ip	ftp	SRC:Auto - DST:SAT(Dmz_FTP_Server)	FTP

Figure 3.9.2 Creating a new SAT policy for the Wi-Fi network

We will use the same FTP settings as in our already existing SAT policy for inbound traffic, meaning we will allow the client to use both active and passive but we will only allow active towards the server.

Recipe 3.10. Configuring Server Load Balancing

Objectives

The objective of this recipe is to discuss and learn how to configure a Server Load Balancing (SLB) IP policy to handle the large amount of requests that is expected to arrive for our university example webpage.

The university webpage is very popular. There are forum discussions, newsgroups, daily updates of schedules, backend systems for school projects and much more. We expect a large amount of requests to be arriving at our webpage. In order to handle the load we will implement an SLB Policy that distributes the traffic between three servers in the DMZ, these servers mirror each other's data.

The benefits of sharing the traffic load across multiple servers can improve not just the performance of applications, but also scalability by facilitating the implementation of a cluster of servers (sometimes referred to as a server farm) that can handle many more requests than just a single server.

The basic principle of SLB functionality is to take requests that arrive on, for example, the EXTERNAL interface IP (External_ip) and then send them to a server based on a distribution algorithm (the algorithm is selected in the SLB Policy).

A representation of what we want to achieve is shown next in *Figure 3.10.1*.

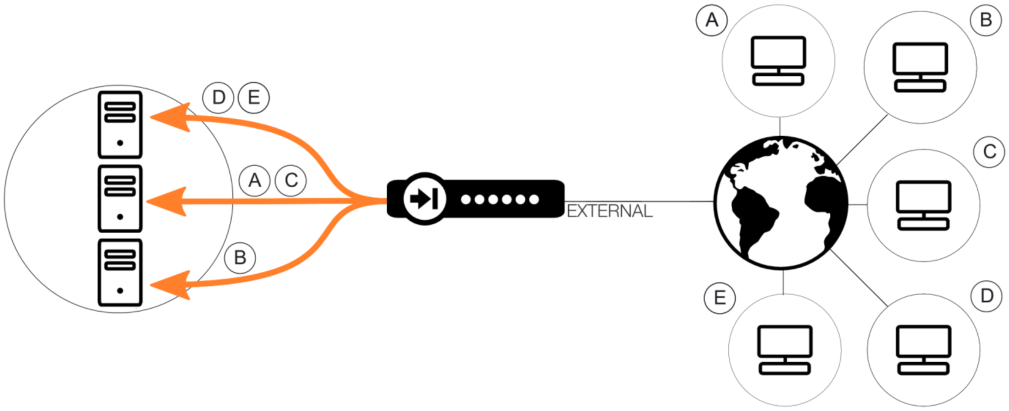


Figure 3.10.1 Server load balancing

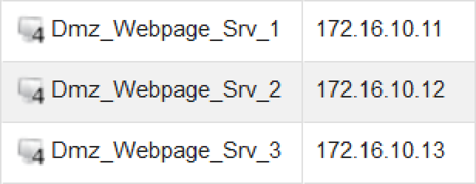
We have multiple clients on the Internet (**A** to **E**) that will connect to the public webpage of the university. When the clients arrive on the external interface we will use an SLB Policy to pick up the traffic and distribute the client requests across our three web servers (located in the DMZ).

Depending on the distribution algorithm chosen, the clients may not always end up on the servers in a sequential order (**A** might not go to server *A*, **B** might not go to server *B*).

Detailed Discussion

Before configuring SLB

Before we create our SLB Policy, we need to create address book objects for our three web servers (in the DMZ), as shown below.



Dmz_Webpage_Srv_1	172.16.10.11
Dmz_Webpage_Srv_2	172.16.10.12
Dmz_Webpage_Srv_3	172.16.10.13

Figure 3.10.2 Address book objects for 3 servers

Next we will create the SLB Policy itself by selecting SLB Policy in the drop down menu.

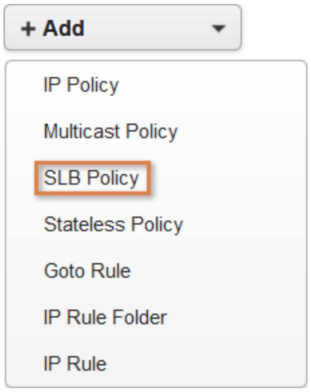


Figure 3.10.3 Creating an SLB Policy

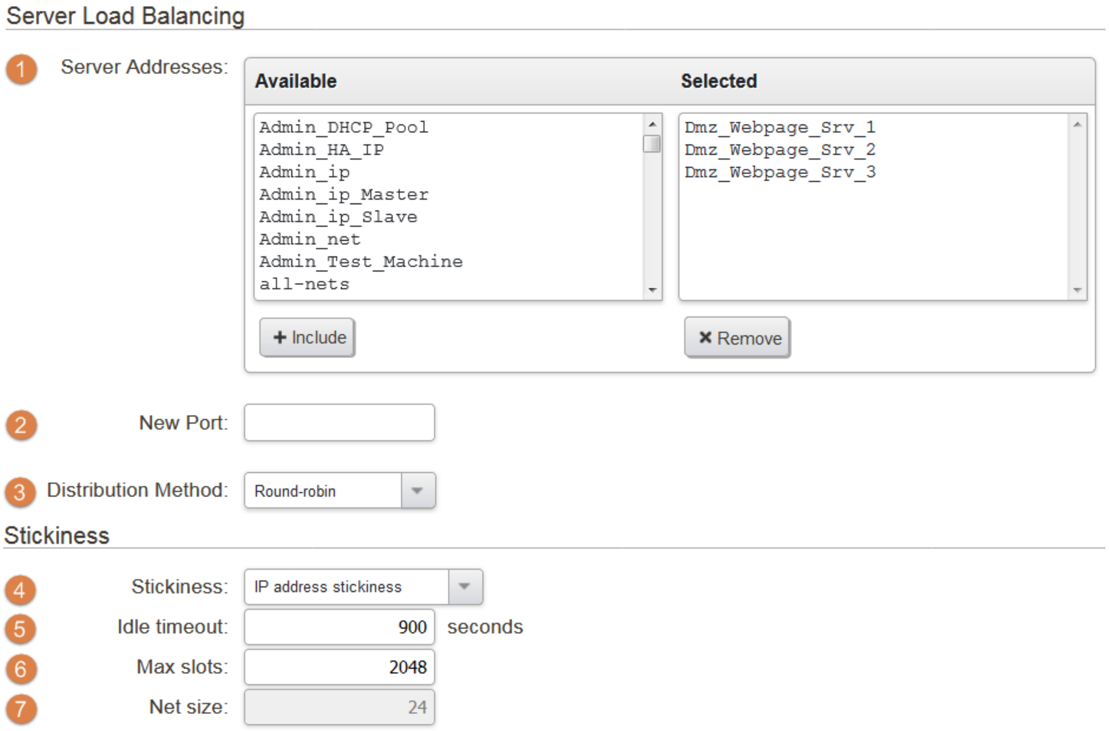
It will be configured in a similar way to the incoming FTP IP policy that was discussed in a previous recipe but in this example we will use http-all as service.

The basic policy properties will be as listed below in *Table 3.10.4*.

Name	Src Iface	Src Net	Dest Iface	Dest Net	Service
Ext_HTTP_ Incoming_SLB	External	all-nets	core	External_ip	http-all

Table 3.10.4   SLB IP policy for the External interface

For SLB policies, we need to go down to the SLB Settings (as shown below) for our new SLB Policy to complete the load balancing configuration, as shown in the screenshot below.



Server Load Balancing

1 Server Addresses:

Available	Selected
Admin_DHCP_Pool Admin_HA_IP Admin_ip Admin_ip_Master Admin_ip_Slave Admin_net Admin_Test_Machine all-nets	Dmz_Webpage_Srv_1 Dmz_Webpage_Srv_2 Dmz_Webpage_Srv_3

2 New Port:

3 Distribution Method: Round-robin

Stickiness

4 Stickiness: IP address stickiness

5 Idle timeout: 900 seconds

6 Max slots: 2048

7 Net size: 24

Figure 3.10.5 SLB Policy settings

The first option we need to select is the server addresses. Here we select our previously created server objects from the address book (**1**). This means users connecting to our public IPv4 address 203.0.113.10 will end up at a server IP address of 172.16.10.11, .12 or .13 after address translation.

Since we are not interested in changing the destination port on the inside, we leave the **New Port** (**2**) option blank. Users going to port 80 will be forwarded to the same port on the inside, only the server IP address will change.

There are several ways to determine how a load is shared across a set of servers. cOS Core SLB supports the following two algorithms for the **Distribution Method** (**3**):

- **Round-Robin** – The algorithm distributes new incoming connections to a list of servers on a rotating basis.
- **Connection-Rate** – This algorithm redirects a connection to the server with the least number of new connections in a given time span.
- **Resource-usage** – This algorithm distributes new incoming connections to a list of servers based on their last reported resource usage.

There is no right or wrong choice here, it will be up to the administrator to decide which method he wants to use based on his requirements. We will use Round-Robin in this example.

Stickiness (**4**) is a something we want to enable. This controls how we should handle multiple requests sent by the same source IP (or network). It will be a good idea to try making sure that requests originating from the same source IP will be sent to the same load balancing server on the inside. The options are none, IP or network.

We will choose IP as even IP stickiness may be problematic if there are many requests coming from the same source IP (for example, when many users are behind NAT). We do not want to overload individual servers in the server farm if we can avoid it.

Idle timeout (**5**) - When a connection is made, the source IP address for the connection is remembered in a table. Each table entry is referred to as a *slot*. After it is created, the entry is only considered valid for the number of seconds specified by the Idle Timeout. When a new connection is made, the table is searched for the same source IP, providing that the table entry has not exceeded its timeout. When a match is found, stickiness ensures that the new connection goes to the same server as previous connections from the same source IP.

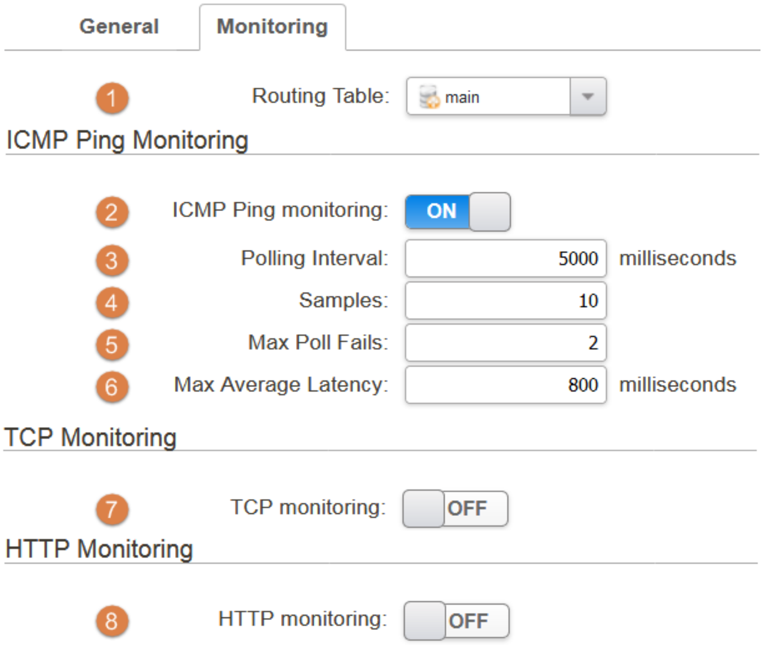
The default value for this setting is 10 seconds but we have set it to 900 (15 minutes) as we want to make sure that a user ends up on the same server as often as possible.

Max slots (**6**) - This parameter specifies how many slots exist in the stickiness table. When the table fills up then the oldest entry is discarded to make way for a new entry even though it may be still valid (the Idle Timeout has not been exceeded). The consequence of a full table can be that stickiness will be lost for any discarded source IP addresses.

The administrator should therefore try to ensure that the Max Slots parameter is set to a value that can accommodate the expected number of connections that require stickiness. Even though we have not changed this from the default it may need to be revised later based on the number of unique users and if they are creating multiple connections towards the server farm.

Server load balancing monitoring

To set up monitoring on our SLB Policy we must go to the Monitoring tab as seen in the next screenshot.



1 Routing Table: main

ICMP Ping Monitoring

2 ICMP Ping monitoring: ON

3 Polling Interval: 5000 milliseconds

4 Samples: 10

5 Max Poll Fails: 2

6 Max Average Latency: 800 milliseconds

TCP Monitoring

7 TCP monitoring: OFF

HTTP Monitoring

8 HTTP monitoring: OFF

Figure 3.10.6 SLB Policy monitoring settings

The **Routing Table** option (**1**) controls which routing table the monitor should use to look for the target hosts. Since we are not using policy based routing or virtual routing in this chapter this will be left as the default value of *main*.

With server farms, it is highly recommended to activate at least one monitoring feature. Since the servers are mirrored it will be no immediate disaster if one server were to go down due for one reason of the other. However, we must be able to detect that it is down and this is what monitoring does. If we do not have any monitoring and one server goes down, we end up in a situation where roughly one third of all requests to our webpage fail.

There are three different types of monitoring available:

- **ICMP** - An ICMP "Ping" message is sent to the server. (**2**)
- **TCP** - A TCP connection is established to the server and then disconnected. (**7**)
- **HTTP** - An HTTP request is sent using a specified URL. (**8**)

Note

The monitoring options will be different depending on which type of monitoring method we choose.

In our example, we will be using the ICMP monitoring method in order to keep things simple. For the ICMP "ping", we have the following monitoring options:

- **Polling interval** (**3**) - This option determines how often cOS Core should send a request to the server(s) to verify whether they are up or not. The default value is 5,000 milliseconds.
- **Samples** (**4**) - Here, we select how large the sample window should be towards the servers. The default value is 10.
- **Max Poll Fails** (**5**) - This option determines how many failed requests in the sample window that we should allow before declaring the server(s) as down. The default value is 2.
- **Max average latency** (**6**) - This option controls the maximum latency we should allow for the response before deciding the request has failed. The default value is 800 milliseconds.

Monitoring Examples

Let's take a few examples on how monitoring works by using the default settings mentioned above. We only use one server as a reference to keep it simple.

Example 1: cOS Core has completed a 10 sample polling request to one of the servers. All responses were within 100 milliseconds so the latency setting never triggered. Since our polling interval is 5,000 milliseconds, it took 50 seconds to perform. The server is considered as up.

Example 2: cOS Core has completed a 10 sample polling request to one of the servers. One of the requests timed out due to unknown reasons, so 1 out of 10 polling samples failed. Since our Max Poll Fails is set to be 2, the server is still considered to be up.

Example 3: cOS Core has completed 6 out of a 10 sample request and all requests have failed to get a reply. Since cOS Core has not completed the 10 sample poll the server is not yet considered to be down even though the Max Poll Fails is set to a value of 2.

Example 4: cOS Core has now completed all 10 sample polling requests from the previous example, and all failed. Since the 10 poll requests are complete cOS core now knows that out of the 10 samples, at least 2 failed. The server is declared as down.

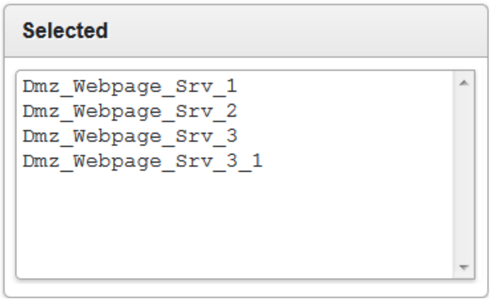
This means that even though we have two failed request within the polling interval, it is not declared as down before the polling window has completed all 10. This means that the time it will take to declare the server as down will be 50 seconds, and not 10.

These settings can be changed to be more aggressive by lowering the polling interval and sample size. However, it will be up to the administrator to decide how aggressive cOS Core should be when performing monitoring.

Sending more requests to more powerful servers

There may be situations where a server farm contains servers of various sizes and processing power. In such situations, there may be the desired behavior that more requests are sent to a particular server with more power than to others with less capability.

There is no direct option to do this, but we can use the distribution algorithms to achieve this. We do this by adding the same server multiple times as shown in figure 3.10.7.



Dmz_Webpage_Srv_1
Dmz_Webpage_Srv_2
Dmz_Webpage_Srv_3
Dmz_Webpage_Srv_3_1

Figure 3.10.7 Srv_3 and Srv_3_1 have the same IP address

In the above example, if we use Round-Robin as distribution algorithm, it will mean that in one full server iteration we will get something like this:

1. A request to **server_1**
2. A request to **server_2**
3. A request to **server_3**
4. A request to **server_3_1**

But since server **3** and **3_1** is the same one, they will get 2 requests effectively sending more traffic to that particular server.

Recipe 3.11. Configuring POP3, IMAP and SMTP

Objectives

The objective of this recipe is to discuss the configuration of the three protocols Post Office Protocol (POP3), Internet Message Access Protocol (IMAP) and Simple Mail Transfer Protocol (SMTP) in cOS Core.

Detailed Discussion

POP3 and IMAP are application-layer Internet standard protocols used by local email clients to retrieve email from a remote server over a TCP/IP connection. Virtually all modern email clients and servers support POP3 and IMAP as they are the two most prevalent Internet standard protocols for email retrieval. Many webmail service providers provide support for either IMAP or POP3 to allow mail to be downloaded from a mail server as shown below in *Figure 3.11.1*.

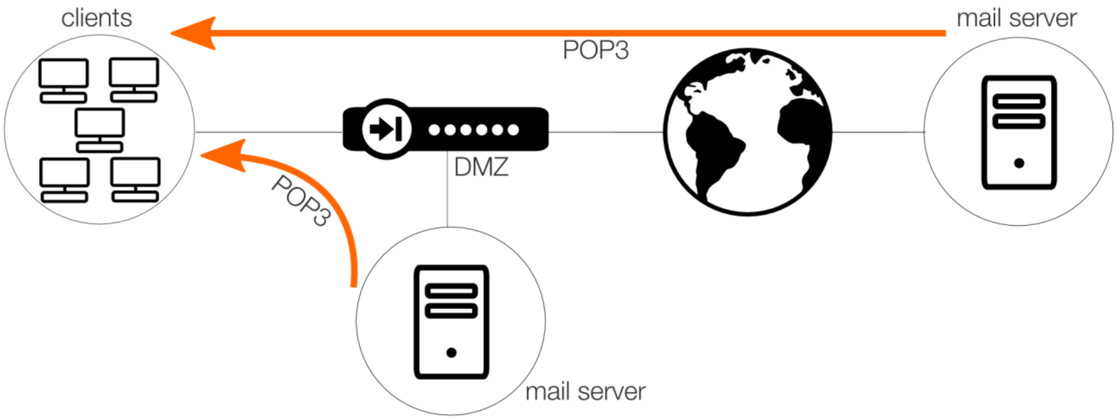


Figure 3.11.1 Example of fetching mail with POP3 from an internal or external server

Simple Mail Transfer Protocol (SMTP) is a text based protocol used for transferring email between mail servers over the Internet. Typically, a local mail server will be located in a DMZ (DeMilitarized Zone) so that mail sent by remote mail servers will traverse the Clavister firewall to reach the local server.

Local clients behind the firewall will then use email client software to retrieve email from the server and to send mail to the server for forwarding out to other mail servers on the public Internet.

Various protocols may be used for communication between clients and a mail server. Retrieval may also be done using POP3 or IMAP as mentioned above, but sending mail to the server may be done using SMTP as shown below in Figure 3.11.2.

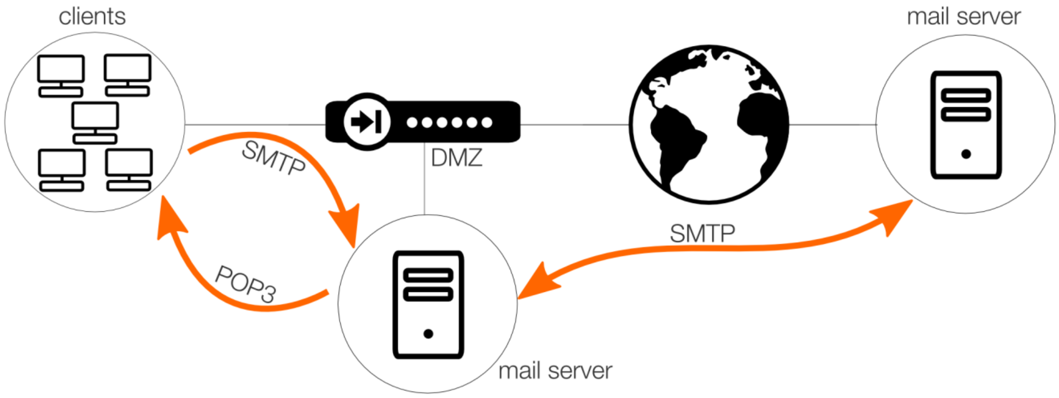


Figure 3.11.2 SMTP and POP3 communication

In order to protect users who use our mail server, we will not only implement our email protocol features for traffic generated from our internal networks to/from the Internet but also between our internal networks and the mail server located in our DMZ zone.

To create an IP Policy where we can use the email protocol features we will make use of Protocols as described earlier in the FTP recipe. First, we create the IP Policy, then we use the service corresponding to the protocol we are using (POP3, IMAP or SMTP). We are then able to apply any profiles related to the supported features, which are Application Control, Anti-Virus, File Control and Email Control.

Configuring Email Control Profiles

The main feature we will talk about regarding the email protocols is Email Control. You can either configure email control directly on an IP policy that has POP3, IMAP or SMTP as protocol or you can create an Email Control Profile that you can then apply to any IP policy with an email protocol defined. We will be using an Email Control Profile so that we don't have to configure the settings manually on every policy where we want to apply Email Control. The Email Control profile can be created under the Policies tab as seen in the next screenshot.

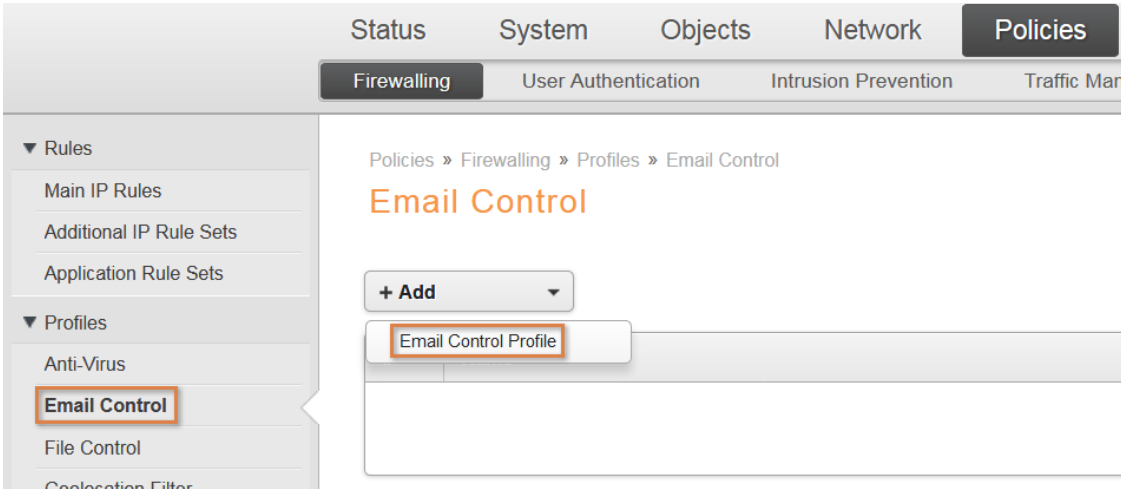
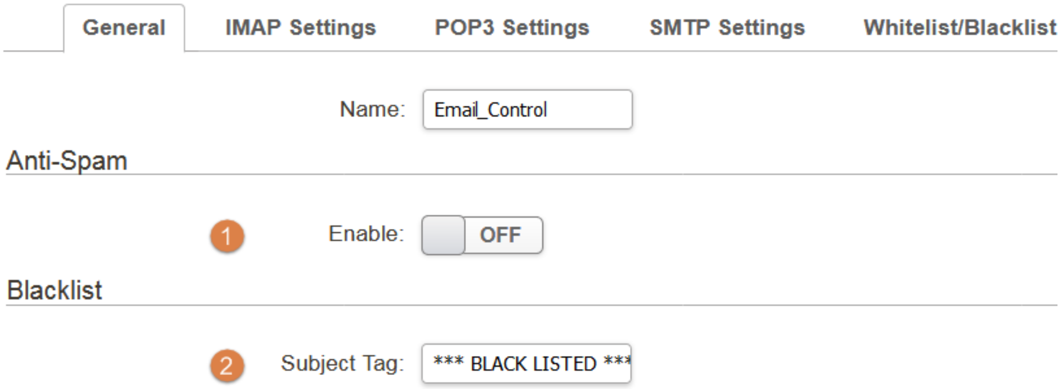


Figure 3.11.3 Email Control Profile location

General Email Control options

First we will go through the options found under the General tab of the Email Control Profile.



Name: Email_Control

1 Enable: OFF

2 Subject Tag: *** BLACK LISTED ***

Figure 3.11.4 General tab options

Anti-Spam (**1**) is one of the major features of the of the Email Control Profile. When enabled, we are met with a bunch of different options to help us define which protection mechanisms should be used to determine if an email should be considered as spam or not. Details regarding these options will be discussed later in this section.

Blacklist Subject Tag (**2**) is the text inserted into the subject field of an email if it is found on the blacklist for this profile. We will go through how to configure Whitelist and Blacklist later in this section.

Spam spam spam spam spam!

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 is 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.

cOS Core offers two approaches to handling spam:

- Dropping email which has a very high probability of being spam. (SMTP only)
- Letting through but flagging email that has a moderate probability of being spam.

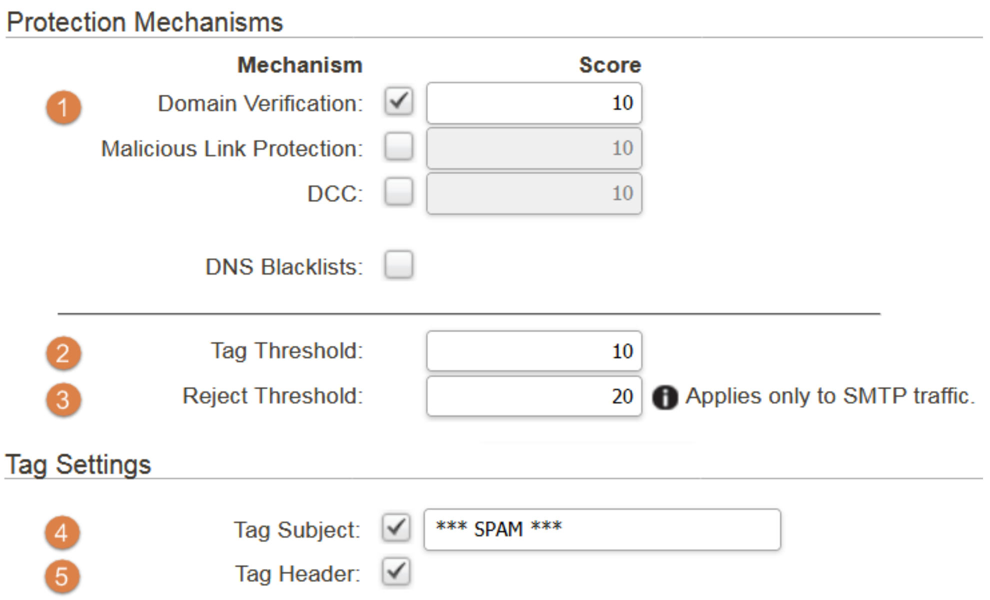
Protection Mechanisms and Filter Scoring

In order to determine if a mail is considered being spam we use something called filter scoring. When an email is received it goes through all enabled protection mechanisms. The total score is calculated by adding together the sub-scores given from the enabled protection mechanisms. If the total score is equal to or greater than the specified Thresholds then the email is considered SPAM and will be either tagged or dropped based on which threshold is reached.

When Anti-Spam has been enabled the administrator is able to choose which protection mechanisms should be used and what amount of score each protection mechanism will add.

The options of the Anti-Spam feature

Since the Anti-Spam feature is such a huge feature with many options, we will split the explanation of its options into multiple images, starting with the those highlighted in the following image.



Protection Mechanisms

Mechanism		Score
Domain Verification:	☑	10
Malicious Link Protection:	□	10
DCC:	□	10
DNS Blacklists:	□	

Tag Threshold: 10

Reject Threshold: 20 Applies only to SMTP traffic.

Tag Settings

Tag Subject: ☑ *** SPAM ***

Tag Header: ☑

Figure 3.11.5 Domain Verification, Thresholds and Tag settings

When **Domain Verification** (**1**) is enabled cOS Core will use DNS to send a mail exchange (MX) record query, requesting the IP address of a mail server associated with the email sender's domain.

If the DNS server recognizes the domain name, cOS Core takes no action. If the DNS server does not recognize the domain, cOS Core will add the configured sub-score for this mechanism to the overall anti-spam score for that email.

Note that at least one DNS server must be configured in cOS Core for this Domain Verification to work. If no DNS server is configured, this test will not be performed and its sub-score will not be added.

Next we will go through the Threshold settings and their functions are quite straight forward.

If the total score for an email is equal to or greater than the specified **Tag Threshold** (**2**) then the email is considered to be SPAM and will be tagged as such.

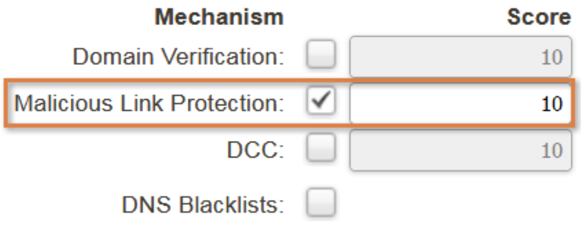
For SMTP only, if the total score for an email is equal to or greater than the specified **Reject Threshold** (**3**) then the email is dropped.

How we tag emails that have reached the Tag Threshold is defined in the Tag Settings.

The **Tag Subject** (**4**) setting adds text to the subject line of an email. The default text is "*** SPAM ***" but this can be set to any string.

The **Tag Header** (**5**) inserts information into the bottom of the email header data which describes the anti-spam processing that has been performed on the email.

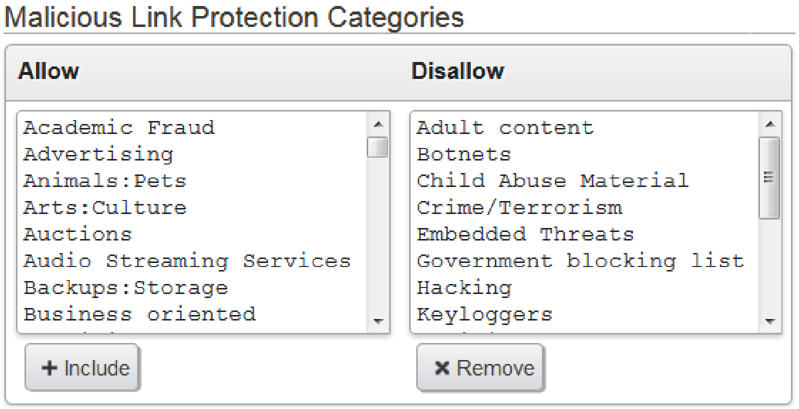
Second we will go through the Malicious Link Protection Mechanism.



Mechanism		Score
Domain Verification:	☐	10
Malicious Link Protection:	☑	10
DCC:	☐	10
DNS Blacklists:	☐	

Figure 3.11.6 Activating Malicious Link Protection

Malicious Link Protection neutralizes undesirable web links inside emails. Finding one or more undesirable links will add the configured sub-score to the total score for that email. Which categories are considered undesirable can be configured with the Malicious Link Protection Categories settings that become available after enabling the Malicious Link Protection mechanism. It's up to the administrator which categories should be allowed or disallowed as seen in the next image.

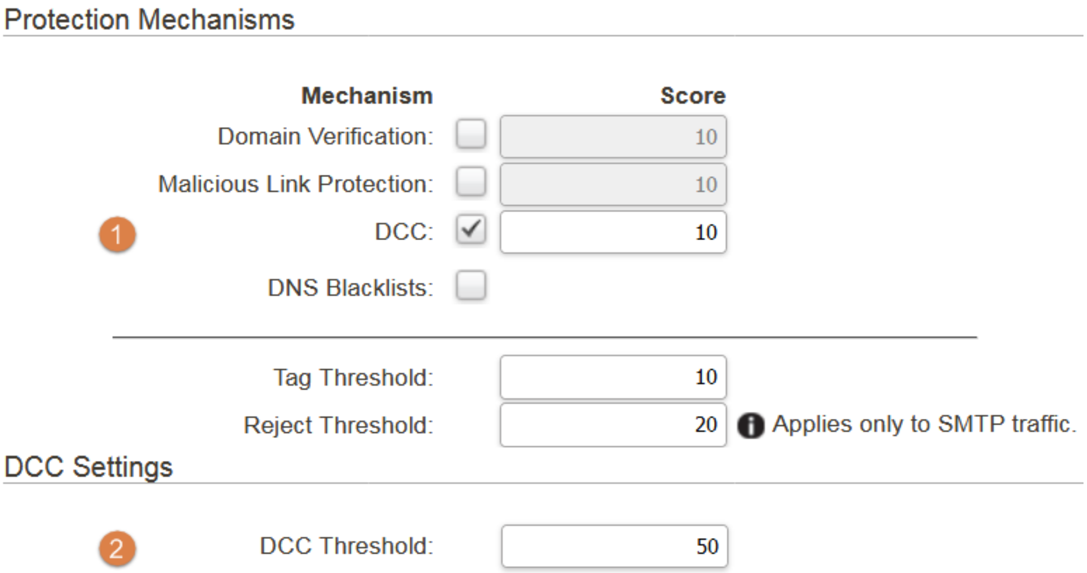


Malicious Link Protection Categories

Allow	Disallow
Academic Fraud	Adult content
Advertising	Botnets
Animals:Pets	Child Abuse Material
Arts:Culture	Crime/Terrorism
Auctions	Embedded Threats
Audio Streaming Services	Government blocking list
Backups:Storage	Hacking
Business oriented	Keyloggers

Figure 3.11.7 Choosing Malicious Link Protection Categories

Next we will look at the Distributed Checksum Clearinghouses (DCC) mechanism. When enabled you will be displayed with the settings available for DCC as seen in the next screenshot.



Protection Mechanisms

Mechanism		Score
Domain Verification:	□	10
Malicious Link Protection:	□	10
DCC:	☑	10
DNS Blacklists:	□	

Tag Threshold: 10

Reject Threshold: 20 Applies only to SMTP traffic.

DCC Settings

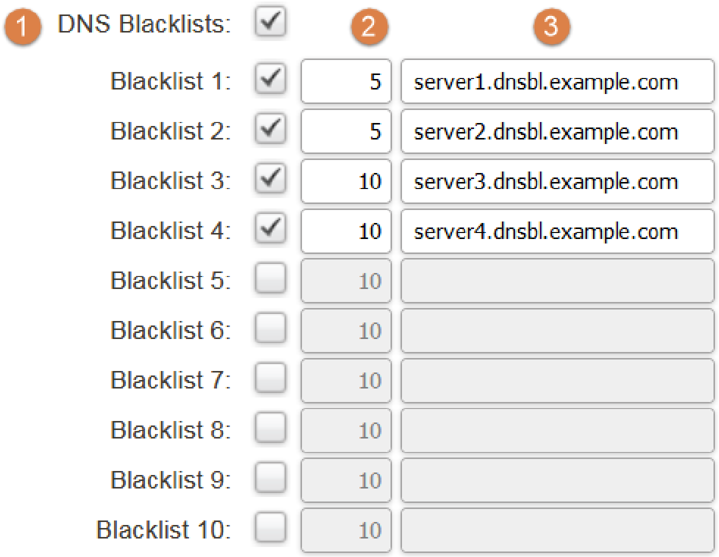
DCC Threshold: 50

Figure 3.11.8 DCC and DCC Settings

Distributed Checksum Clearinghouses (**DCC**) (**1**) is an external server network that acts as a central repository for email checksums reported by participating email servers. cOS Core sends the email checksum of the received email to the Clearing House server and if the returned value is greater than the defined **DCC Threshold** (**2**), the sub-score for this mechanism will be added to the total anti-spam score.

The idea is that if a checksum has been reported many times by many different servers then it is probably spam.

The final option we have left to discuss is DNS Blacklists.



DNS Blacklists:	☑		
Blacklist 1:	☑	5	server1.dnsbl.example.com
Blacklist 2:	☑	5	server2.dnsbl.example.com
Blacklist 3:	☑	10	server3.dnsbl.example.com
Blacklist 4:	☑	10	server4.dnsbl.example.com
Blacklist 5:	☐	10	
Blacklist 6:	☐	10	
Blacklist 7:	☐	10	
Blacklist 8:	☐	10	
Blacklist 9:	☐	10	
Blacklist 10:	☐	10	

Figure 3.11.9 Using DNS Blacklists

By enabling the DNS Blacklists (DNSBL) option (**1**), we are able to define up to 10 different DNS Blacklist (DNSBL) servers can be specified (**3**), each with its own sub-score (**2**)(the default value is 10 per server) that will be added to the total score if that server flags an email as spam. If a DNSBL server is not responding then its sub-score is not included in the final score calculation.

In this example we have four servers, two with value 5 and two with value 10. These values are set by the administrator based on how trustworthy the response from this particular server is. We trust DNSBL servers 1 and 2 only half as much as DNSBL servers 3 and 4.

DNS Blacklist Databases

A number of trusted organizations maintain publicly available databases that contain the origin IP address of known spamming SMTP servers and these can be queried over the public Internet. These databases are known as DNS BlackList (DNSBL) databases and the information is accessible using a standardized query method which is supported by cOS Core.

When the cOS Core Anti-Spam filtering function is configured, the IP address of the email's sending server is sent to our configured DNSBL servers to find out if any of the DNSBL servers think the email is from a spammer or not. The reply sent back by a DNSBL server is either a non-listed response or a listed response. In the latter case of being listed, the DNSBL server is indicating that 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.

Note

DNSBL servers do not blacklist individual email addresses but rather mail server IPs.

The schematic below illustrates DNSBL processing. If we for example would configure two DNSBL servers, cOS Core will query the two DNSBL servers and ask them if the sender of an incoming mail is a potential risk or not and, based on the answer, will apply the configured sub-score for that server.

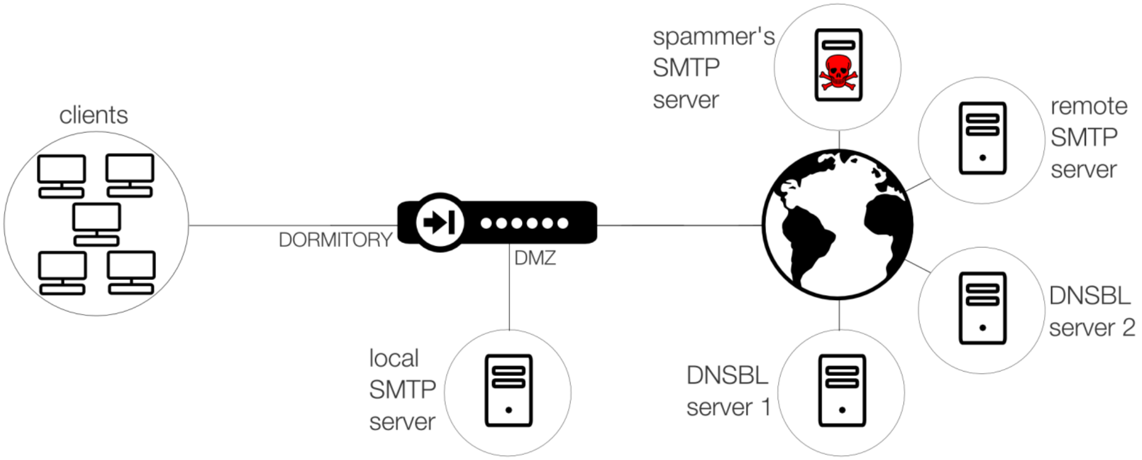
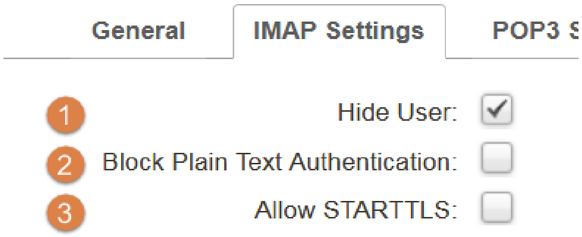


Figure 3.11.10 Using DNSBL for Anti-Spam

Since we want to be as effective as possible in detecting spam we will enable all Protection Mechanisms available for our email traffic and leave them with their default settings. The only thing that has to be defined by the administrator is which DNSBL servers should be used.

The IMAP Settings in Email Control

The options for found under the IMAP settings tab are relatively few. The options can be seen in figure 3.11.11.



1 Hide User: ☑
2 Block Plain Text Authentication: □
3 Allow STARTTLS: □

Figure 3.11.11 IMAP Settings

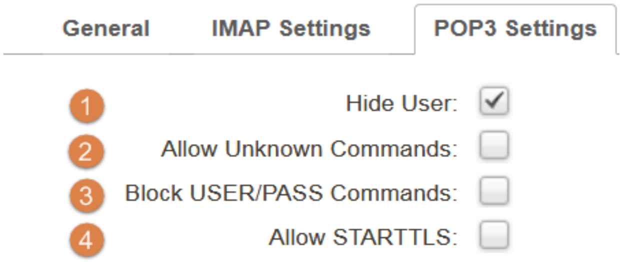
The first option (**1**) prevents the IMAP server from revealing that a username does not exist. This prevents users from trying different usernames until they find a valid one. We will enable this option.

Block Plain Text Authentication (**2**) enables us to block connections between client and server that send the username/password combination as easily-read clear text (some servers may not support other methods than this). We will allow this to be as compatible as possible but it can easily be changed if needed. If there are no servers that only support Plain Text Authentication, it is recommended to enable this.

When the **Allow STARTTLS** (**3**) option is enabled, clients are allowed to use the STARTTLS command which means that we are allowing clients to switch to using encrypted mail transactions. This prevents any protection mechanisms or anti-virus scans to be done on the emails as we are unable to scan the encrypted data. Because of this we will keep this option disabled.

The POP3 Settings in Email Control

POP3 has some options that works the same way as the ones for IMAP as well as some unique options for POP3 as seen in the next screenshot.



1 Hide User: ☑
2 Allow Unknown Commands: □
3 Block USER/PASS Commands: □
4 Allow STARTTLS: □

Figure 3.11.12 POP3 Settings

The first option **Hide User** (**1**) works the same way as it does under IMAP settings, which means that it prevents the POP3 server from revealing that a username does not exist.

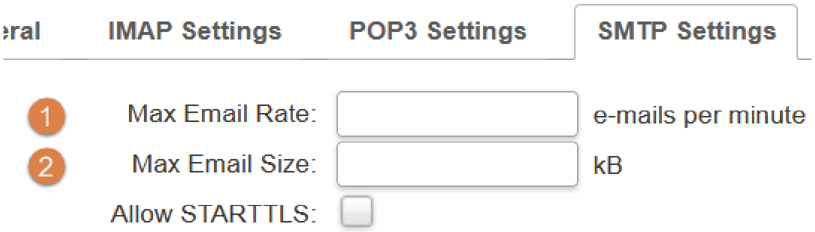
Allow Unknown Commands (**2**) is an option that allows clients to use commands that are not standard POP 3 commands. We will not allow this but it can be easily changed if needed.

The option **Block USER/PASS Commands** (**3**) has a similar function as the Block Plain Text Authentication option in IMAP. It lets us block clients from sending their credentials in plain text using the USER and PASS commands.

The Allow STARTTLS (**4**) option has the same function as described in IMAP settings. It allows clients to use encrypted mail transactions but we won't be able to scan the emails for viruses or spam.

The SMTP Settings in Email Control

There are two main options that are of interest here, the Allow STARTTLS option for SMTP has the same function as explained in the previous two protocols.



Max Email Rate: e-mails per minute

Max Email Size: kB

Allow STARTTLS: □

Figure 3.11.13 SMTP Settings

The first is the **Max Email Rate** (**1**). This option controls how many mails are allowed to be sent from the same host within 60 seconds. We leave this option blank as we do not know the exact email rate that we will encounter and initially we do not want to restrict our users until we know the traffic pattern. What is a good value? It all depends on the size of the network and the traffic pattern, 10? 100? It will be up to the administrator to set an appropriate level for his environment.

The second option, **Max Email Size** (**2**), controls the maximum size (in kilobytes) of the emails being sent through the SMTP ALG. This option can be good to use if we want to avoid users sending emails with large file attachments. Once again, we leave this option blank in our example as it all depends on the network requirements and whether the administrator wants to restrict this or not.

Using Whitelists and Blacklists

We can add a whitelist or a blacklist for an IP by going to the Whitelist/Blacklist tab and clicking add Email Filter as seen in the next screenshot.

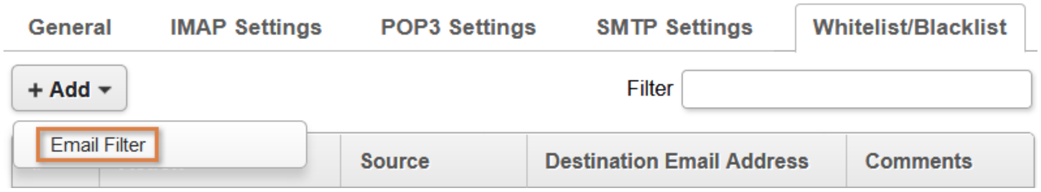
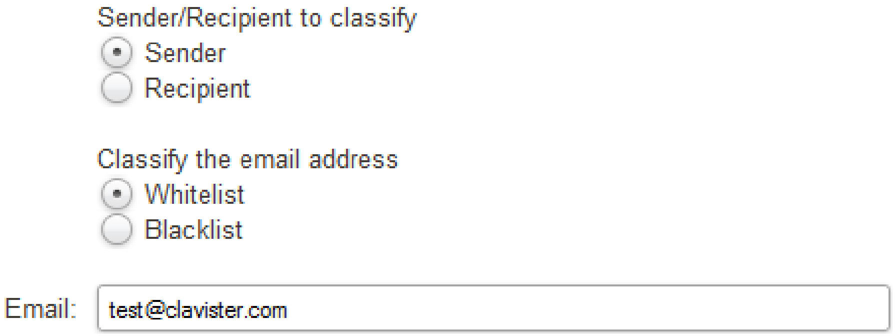


Figure 3.11.14 Location of adding Whitelists or Blacklists

Whitelist/Blacklist is a function that can be used to avoid a user accidentally getting blacklisted (as shown below). For instance, we do not want the administrator to get blacklisted by mistake. To make sure that never happens we can put that user into the whitelist. Or, if we have a user that we know is infected by a virus or otherwise compromised, we can add them into the blacklist.



Sender/Recipient to classify
● Sender
○ Recipient

Classify the email address
● Whitelist
○ Blacklist

Email: test@clavister.com

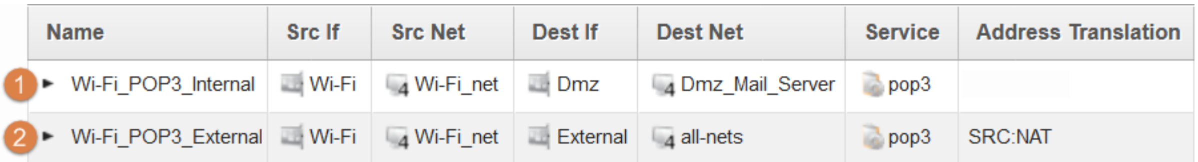
Figure 3.11.15 Whitelisting and blacklisting

Wildcards can be used here, for instance, the list entry **@clavister.com* can be used to specify all possible email addresses related to the domain.

POP3 IP Policy configuration

Now that we have created our Email Control profile, all that's left is for us to set up the Policies for the email traffic, beginning with POP3.

For POP3, we will create two IP policies for each interface involved, as shown next.



Name	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
Wi-Fi_POP3_Internal	Wi-Fi	Wi-Fi_net	Dmz	Dmz_Mail_Server	pop3	
Wi-Fi_POP3_External	Wi-Fi	Wi-Fi_net	External	all-nets	pop3	SRC:NAT

Figure 3.11.16 POP3 IP policies

These policies are repeated for the interfaces ADMIN, Wi-Fi, TEACHERS and DORMITORY.

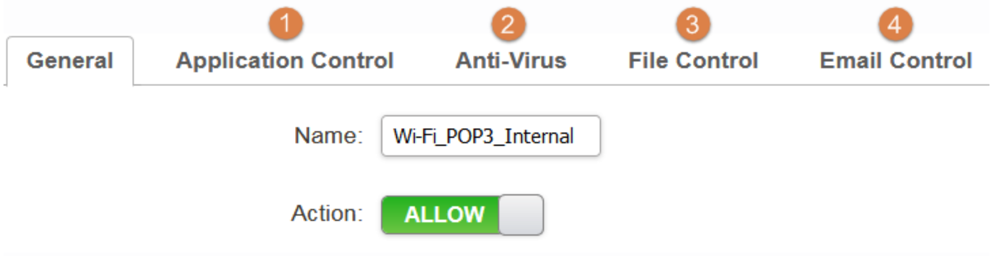
The first policy (**1**) allows communication between the client network and the internal POP3 server in the DMZ zone. We do not need any address translation for this policy as it is communication between two internal networks and we have no reason to mask the client's source IP.

It will also be advantageous to be able to see the client's source IP when troubleshooting any problems using the logs and for this reason one should avoid address translation if possible.

The second policy (**2**) allows communication between the client network and a host/server on the Internet. It is reasonable to assume that many users have mail accounts on other servers, not just the university. Since we want to go from a private IP addresses towards public IP addresses we have to enable NAT as source translation. Both of these two policies could be using Auto as source translation as well without any issues as it would be able to detect whether the traffic needs to be NAT'ed or not.

Using email specific features

Earlier in this recipe it was mentioned that POP3, as well as IMAP and SMTP supports the following four features, Application Control, Anti-Virus, File Control and Email Control. To be able to use these features it is important that the service we have selected on our IP Policies is using POP3, IMAP or SMTP as protocol. Therefore we will use the default pop3 service for our POP3 policies as it is preconfigured with POP3 as protocol. We will now go through how these four features can be used in relation to our email protocols.



Name: Wi-Fi_POP3_Internal

Action: ALLOW

Figure 3.11.17 Email specific features

As seen in the screenshot above, the first feature is **Application Control** (**1**). This feature lets us control things like which email clients our users are allowed to use as well as well as more detailed control over what content we are allowed to have in our emails in terms of attachments and various other things. We will not be using Application Control on any of our email protocols as it depends on the requirements of the network and at what level of control the administrator wants to have. For more details regarding Application Control, see the next upcoming recipe *Recipe 3.12. Using Application Control*

We have discussed **Anti-Virus** (**2**) several times in other recipes so we will not go into details regarding the Anti-Virus settings. Please see *Recipe 3.7. Configuring Anti-Virus protection* for more details about Anti-Virus configuration. We will apply an Anti-Virus profile set to protect mode on both of our policies for each interface. We can reuse the Anti-Virus profile created in earlier recipes for this or we could create a new one to be used. We will be reusing our previously configured Anti-Virus profile.

File Control (**3**) is a function that can be used to block specific file types such as .exe or .zip. For now we will not be using it.

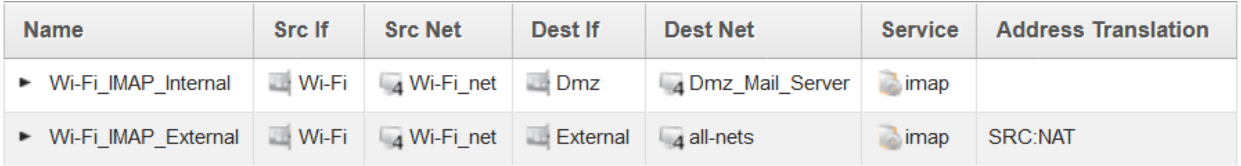
Email Control (**4**) is the tab where we will assign our previously created Email Control Profile. We will assign our Email Control Profile to all created policies, both internal and external.

Note

It is not uncommon for mail servers to use SSL or similar encrypted protocols for POP3 and SMTP traffic. None of these features support encrypted traffic, so they cannot be used for such encrypted mail.

IMAP IP Policy configuration

Our IP Policy configuration for IMAP will be very much like the configuration we have set up for POP3. We will set up one policy to allow traffic between the client network and a host/server on the Internet with the IMAP protocol. Then we will set up a second policy to allow IMAP traffic towards our internal mailserver since it supports mail fetching via both IMAP and POP3.



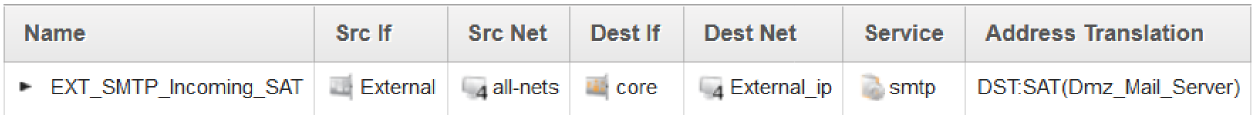
Name	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
► Wi-Fi_IMAP_Internal	Wi-Fi	Wi-Fi_net	Dmz	Dmz_Mail_Server	imap	
► Wi-Fi_IMAP_External	Wi-Fi	Wi-Fi_net	External	all-nets	imap	SRC:NAT

Figure 3.11.18 IMAP IP policies

On these two policies we will make sure to select the default imap service as it is preconfigured with IMAP as protocol. We can then assign our already configured Anti-Virus profile as well as our Email Control profile to these policies.

SMTP IP Policy configuration

As we want to protect our mail server in the DMZ, we use our Anti-Virus and Email Control profiles with a SAT IP policy using the default smtp service object as service as it is preconfigured with SMTP as protocol. This allows incoming connections from the Internet to our web server in the DMZ as shown in the policy shown below.



Name	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
► EXT_SMTP_Incoming_SAT	External	all-nets	core	External_ip	smtp	DST:SAT(Dmz_Mail_Server)

Figure 3.11.19 SMTP SAT Policy

The processing order used in email filtering

Email filtering obeys the following processing order and this is similar to the order followed by the Web Profile except for the addition of Spam filtering:

1. Whitelist.
2. Blacklist.

3. Spam filtering (If enabled with Email Control Profile).

4. Anti-virus scanning (If enabled with Anti-Virus Profile).

This means that if a sender address is both in the blacklist and the whitelist, the whitelist will have precedence, so a whitelisted user will never have any problems sending or receiving email.

Recipe 3.12. Using Application Control

Objectives

Up until now, we have been controlling access levels using ports and to some extent application layer gateways. However, there are many scenarios where multiple applications are using the same port or protocol. So how can we block one but allow the other if they are using the same port? The objective of this recipe is to describe the configuration and use of Application Control (AC) to decide which programs should be allowed network access.

Detailed Discussion

In the example illustrated below in *Figure 3.12.1*, we want to allow users to be able to connect to Facebook but not Youtube.

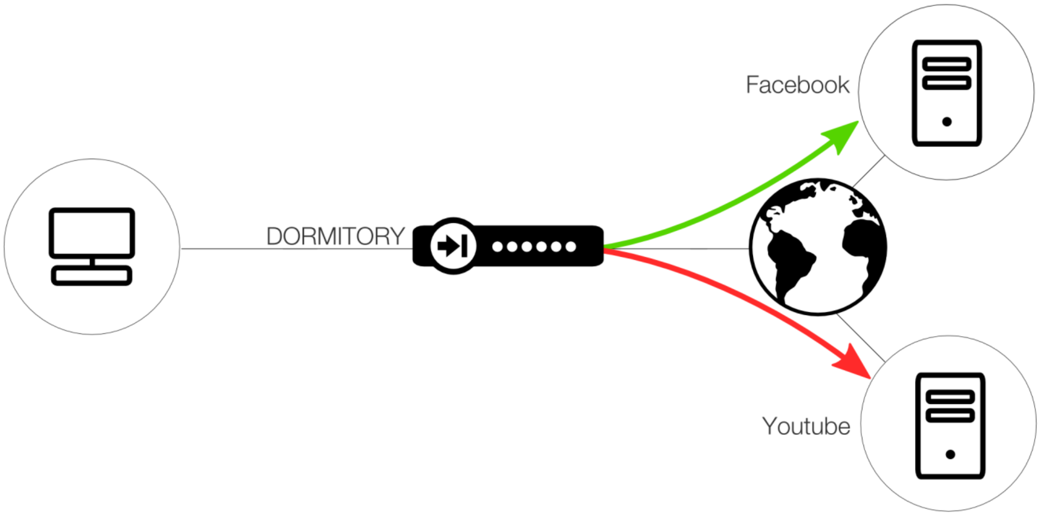


Figure 3.12.1 Allowing access to Facebook but not Youtube

By using application control, we can look at the traffic/packets being generated by our users and match that against an application signature database. Each signature corresponds to one type of application.

To accomplish this, we will first create an Application Rule Set that determines how application control should behave and which applications we want to allow or deny. We can also use and create Application Control settings directly on an IP policy, but we will create an Application Rule Set as we want to reuse the same AC rule set on multiple policies. Creating such rule sets is done under the policies section, as shown in the next screenshot.

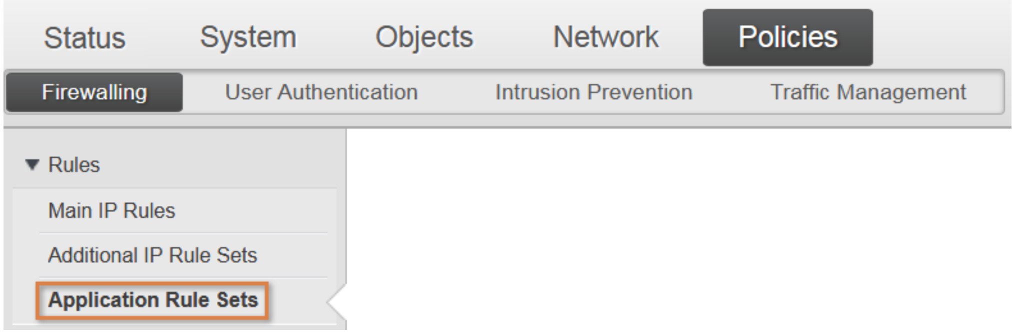


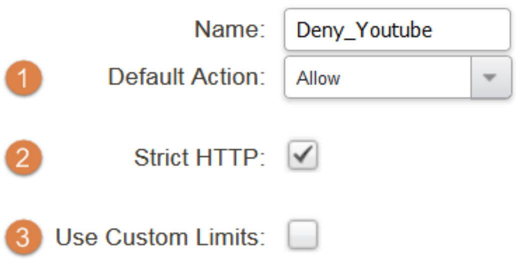
Figure 3.12.2 The location of the Application Rule Sets

Note

We will use Youtube and Facebook as examples here, but in the end it will be up to the administrator to decide what applications to allow or deny. Some admins only want to log what applications are used.

The importance of understanding the default action option

When creating an Application Rule Set, we are first presented with some general options, as shown in the next WebUI screenshot.



Name: Deny_Youtube

Default Action: Allow

Strict HTTP: ☑

Use Custom Limits: □

Figure 3.12.3 General Application Rule Set options

The **Default Action** (**1**) is a very important setting. It can decide how AC will handle applications that match our filter criteria (if any).

The two possible values for the Default Action are:

- **Allow** - When in allow mode, all applications will be allowed except the specific applications we chose to deny in our AC rule filters. For example, deny Youtube, allow all other applications.
- **Deny** - When in deny mode, all applications will be denied except the specific applications we choose to allow in our AC rule filters. For example, allow Facebook and DNS, deny all other applications.

Note

It should be highlighted again how important it is to understand the difference in these two actions. They will make AC function very differently based on the choice.

The **Strict HTTP** (**2**) option determines how aggressive cOS Core should be when it comes to the handling of plain HTTP as many programs use the HTTP port not just for HTTP data. This setting makes cOS Core look harder at HTTP data to try determine what it is.

Note: Additional information about Strict HTTP

Many protocols that application control examines are built on top of the HTTP protocol. In some cases where HTTP itself is being blocked by application control, a protocol built on HTTP may be erroneously blocked as well. To try to resolve this problem, the Strict HTTP setting can be disabled for the relevant Application Rules. This will force application control to evaluate the entire protocol structure before making a decision on the protocol type.

The **Custom Limit** (**3**) setting allows us to override the global advanced setting that controls the limits on how much we should scan a packet before we classify it as "unknown". The maximum amount of data processed to make this decision is specified in cOS Core as both a number of packets and a number of bytes.

By default, these two values are the following:

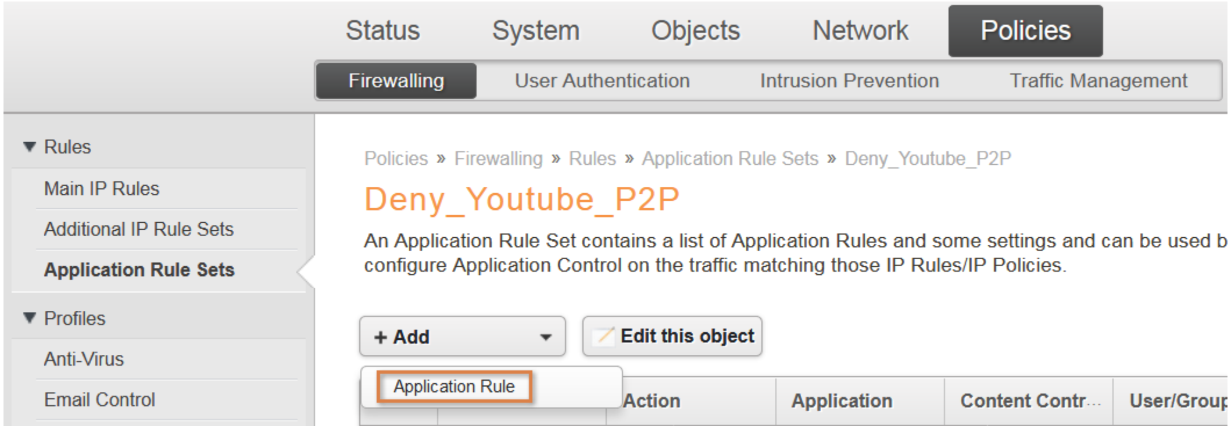
- Max Unclassified Packets: 5
- Max Unclassified Bytes: 7,500

When either of these values is reached, the unclassifiable decision is made. If the administrator wants to make individual changes to these settings without making it a global change, it can be done here. We will leave this setting as default.

These options can be changed globally under **System -> Advanced Settings-> Application Control Settings**.

An AC example: Deny Youtube, allow other applications

In our first example we want to create a new application rule set that specifically blocks access to Youtube and Peer to Peer File sharing but allows all other applications. After creating the new application rule set we go into the rule set and add a new application rule as seen in the next screenshot.



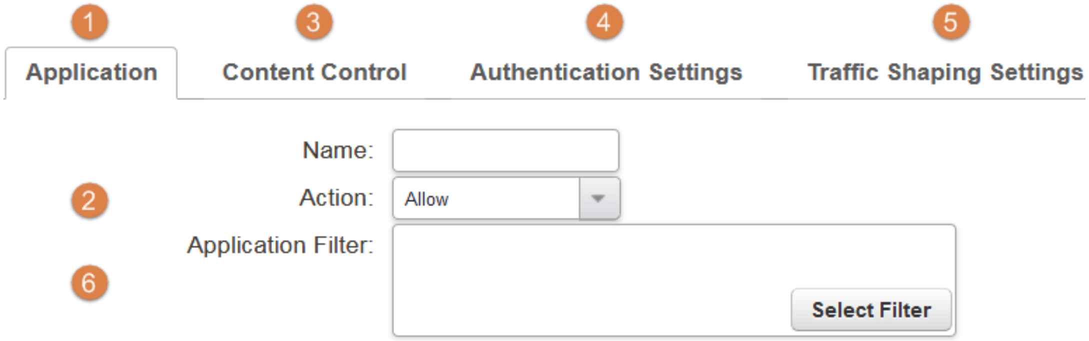
Policies » Firewalling » Rules » Application Rule Sets » Deny_Youtube_P2P

Deny_Youtube_P2P

An Application Rule Set contains a list of Application Rules and some settings and can be used b configure Application Control on the traffic matching those IP Rules/IP Policies.

Figure 3.12.4 Adding a new Application Rule

Now, we have arrived at the main settings and options for Application Control, as shown below. We will primarily concentrate our efforts on the **Application** (**1**) tab where we select which applications should be allowed or denied based on the **Action** (**2**) (Allow or Deny). Before we do that, let us mention some of the other options we have.

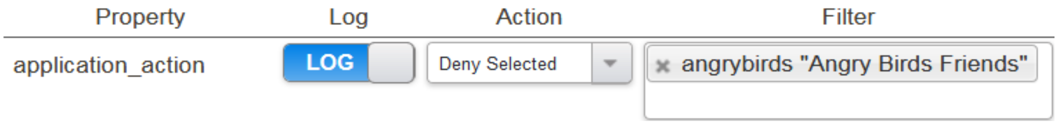


Name:	
Action:	Allow
Application Filter:	

Figure 3.12.5 Application Control options and tabs

Content Control (**3**) provides additional filtering detail based on which application(s) we have selected (**6**).

Taking Facebook as an example, the next screenshot gives an idea of what can be filtered within Facebook using Content Control.



Property	Log	Action	Filter
application_action	LOG	Deny Selected	angrybirds "Angry Birds Friends"

Figure 3.12.6 Filtering with Content Control

Content control gives us the ability to control applications within other applications. We will not be using Content Control in this example as it depends heavily on the requirements of the network and at what level of detail the administrator wants to control applications. It can be made as simple or as complex as the administrator wants.

With **Authentication Settings** (**4**) we can configure the rule to only trigger for specific users or groups. For an **Allow** rule, the requesting client is only permitted the connection if they have already been authenticated by cOS Core and are one of the usernames specified for the rule or belong to one of the specified groups.

For a **Deny** rule, the requesting client is denied the connection if they are authenticated and are one of the usernames specified or belong to one of the specified groups. Authentication may have been performed using any of the methods available in cOS Core Authentication Rule objects, including Identity Awareness. If no groups or usernames are specified in the rule, authentication is ignored.

We will go into more details regarding authentication in the next chapter.

Traffic Shaping Settings (**5**) - Traffic Shaping is a feature where we can control how much bandwidth a set of applications is allowed to use. It can be very useful to prioritize bandwidth between different applications and, if used in conjunction with the authentication setting, on particular users or user groups using that application.

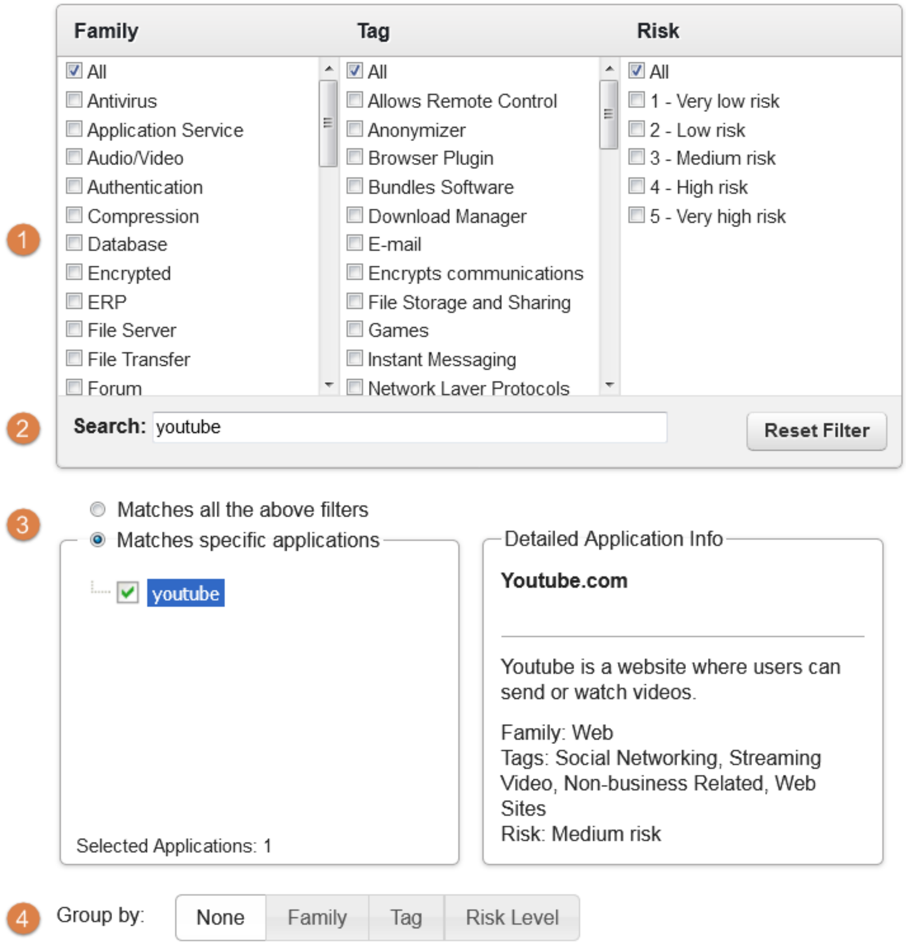
Traffic shaping is only applicable if the Application Rule has an action of **Allow**.

We will discuss and use traffic shaping in much greater detail in the next chapter. For now, we will not activate this particular feature.

Selecting applications when the default action is "Allow"

When our Application Rule Set is created, we need to select which applications we want to allow or deny. Since in our current example we are using default action "Allow", this means that unless we specifically add applications to deny, everything will be allowed.

When selecting which applications we want to add to our filter, we click the **Select Filter** button shown and will be presented with the filter options shown in the next screenshot.



Family

- ☑ All
- □ Antivirus
- □ Application Service
- □ Audio/Video
- □ Authentication
- □ Compression
- □ Database
- □ Encrypted
- □ ERP
- □ File Server
- □ File Transfer
- □ Forum

Tag

- ☑ All
- □ Allows Remote Control
- □ Anonymizer
- □ Browser Plugin
- □ Bundles Software
- □ Download Manager
- □ E-mail
- □ Encrypts communications
- □ File Storage and Sharing
- □ Games
- □ Instant Messaging
- □ Network Layer Protocols

Risk

- ☑ All
- □ 1 - Very low risk
- □ 2 - Low risk
- □ 3 - Medium risk
- □ 4 - High risk
- □ 5 - Very high risk

Search: youtube

Reset Filter

○ Matches all the above filters
● Matches specific applications

☑ youtube

Selected Applications: 1

Detailed Application Info

Youtube.com

Youtube is a website where users can send or watch videos.

Family: Web
Tags: Social Networking, Streaming Video, Non-business Related, Web Sites
Risk: Medium risk

Group by: None | Family | Tag | Risk Level

Figure 3.12.7 The application filter view with options

The Application Control database consists of more than 2000 signatures, making it a requirement to be able to narrow down what we are looking for. To accomplish this we have several tools available.

From the first options (**1**) we can narrow down our search or simply select a particular **Family**, **Tag** or **Risk level**. We can, for instance, choose to deny the entire *Very High Risk* group if we want.

With our second option (**2**) we can use a string to narrow down the search. In our example, we have chosen to search for "youtube". This yielded only one hit, which is exactly what we were looking for.

The third option (**3**) determines how we want to select the applications that match our filters. There are two choices:

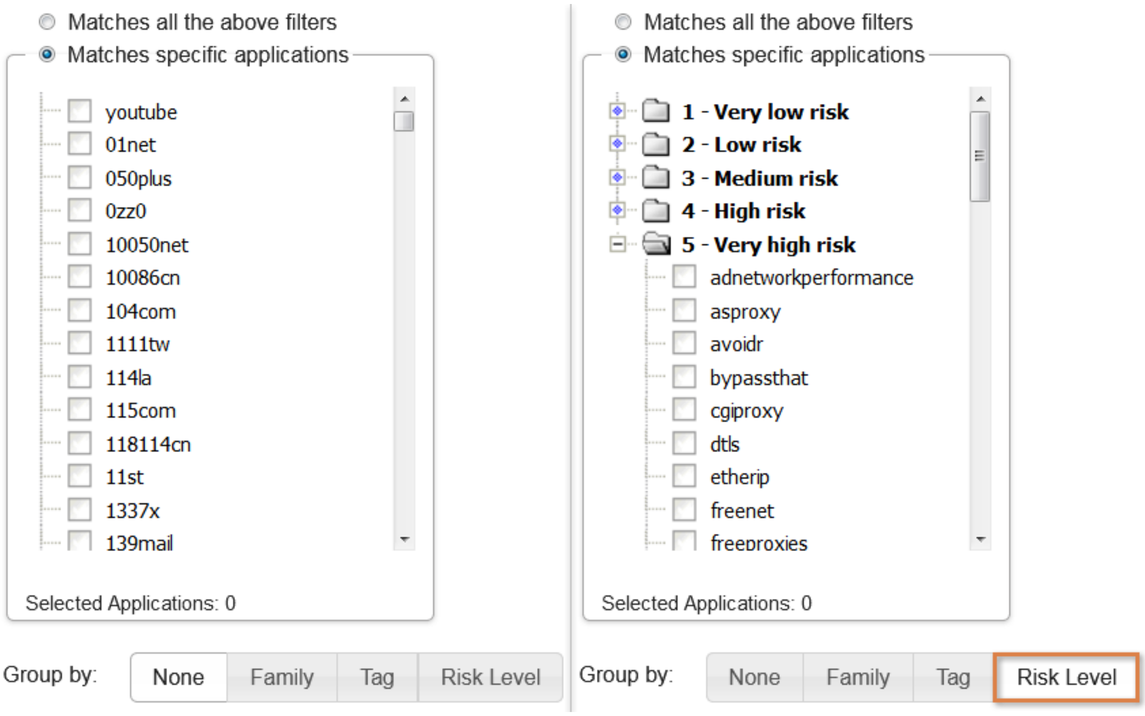
- **Matches all the above filters**

  This option means that applications will be based on our chosen Family, Tag and Risk group. This does NOT include the search field. It is not possible to make a matching filter based on a search string.

- **Matches specific applications**

  This option means that the summary view of target applications changes so that we will get the option to enable/disable specific applications using a checkbox. For instance the search result may yield 3 applications, and we only want to use one. Then we can use this option and only select the one application that best matches our filters by enabling its checkbox (**3**).

The fourth and last option (**4**) allows us to change the groupings of applications. By default there is no grouping but by changing grouping to, for example, **Risk Level** (as seen in the next screenshots) we can get a different kind of filter view in order to more easily find what we are looking for.



○ Matches all the above filters
● Matches specific applications

- □ youtube
- □ 01net
- □ 050plus
- □ 0zz0
- □ 10050net
- □ 10086cn
- □ 104com
- □ 1111tw
- □ 114la
- □ 115com
- □ 118114cn
- □ 11st
- □ 1337x
- □ 139mail

Selected Applications: 0

Group by: None | Family | Tag | Risk Level

○ Matches all the above filters
● Matches specific applications

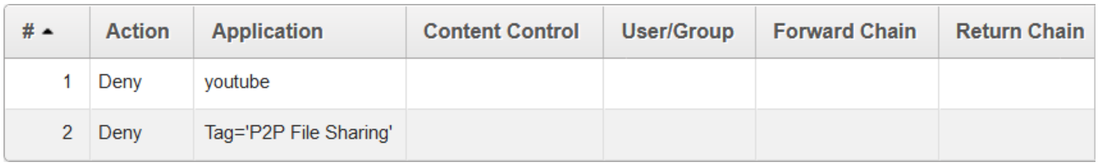
- 1 - Very low risk
- 2 - Low risk
- 3 - Medium risk
- 4 - High risk
- 5 - Very high risk
  - □ adnetworkperformance
  - □ asproxy
  - □ avoidr
  - □ bypassthat
  - □ cgiproxy
  - □ dtls
  - □ etherip
  - □ freenet
  - □ freeproxies

Selected Applications: 0

Group by: None | Family | Tag | Risk Level

Figure 3.12.8 Using Risk Level grouping and no application grouping

In order to keep the examples simple, we have chosen to only deny two filter types. This is shown in the following WebUI screenshot.



# ▲	Action	Application	Content Control	User/Group	Forward Chain	Return Chain
1	Deny	youtube				
2	Deny	Tag='P2P File Sharing'				

Figure 3.12.9 The "Default_Allow" Application Rule Set filters

For our first **Deny** rule, we used the search function to find the youtube application, then we changed the matching type to unlock the "checkbox" method of selecting applications. By doing that we narrowed down our search to only one application.

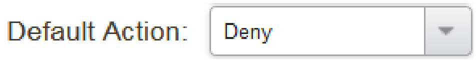
For our second **Deny** rule, we used the default matching type to select a specific *Tag* group, in our case the "Peer to Peer File sharing" tag group. We add the P2P File Sharing Tag to our filter in this example to demonstrate the way you can use Tag groups in application control.

Note

It is recommended to go through the resulting applications when selecting a Family, Tag or Risk level. In our example we chose to deny the "P2P File Sharing" tag group but that group also contains applications that we may want to allow in our network, such as Spotify.

Selecting applications when the default action is "Deny"

Previously, we have used an Application Rule Set where the default action was **Allow**. This means that all applications, unless specifically blocked, will be allowed. This time, we are going to do the reverse.



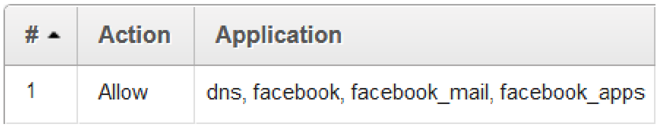
Default Action: Deny

Figure 3.12.10 Setting the Application Rule Set action to Deny

By setting the default action to **Deny**, as shown above, everything will be blocked except the applications we chose to allow.

This has the big advantage that if there is unknown application in the network that is not identifiable by AC, they will automatically be disallowed. It can naturally also be a drawback, but generally speaking it is best to deny an unknown application first until we know what it is. Then after wards, once we know what it is either keep denying it or make an exception for it in the rules.

As a simple example, we create a rule set that only allows a very limited amount of applications, in our case only DNS and Facebook, as shown in the following screenshot.



# ▲	Action	Application
1	Allow	dns, facebook, facebook_mail, facebook_apps

Figure 3.12.11 The allow list for the "Default=Deny" rule set

This means that if we use this Application Rule Set, our user(s) would only be able to resolve DNS names and connect to Facebook. All other applications would be denied.

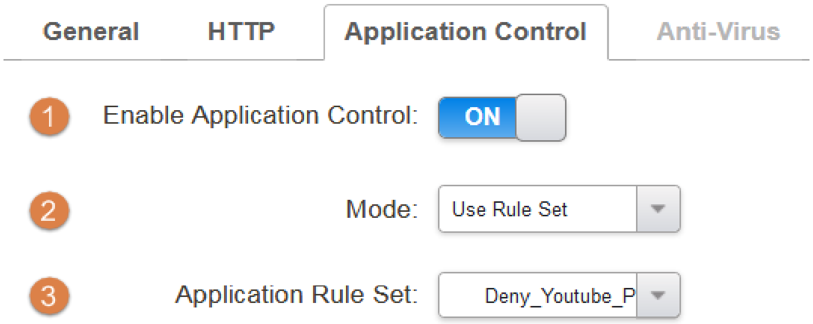
Note

It should be highlighted how important it is to understand the differences with the default action, as it will radically change the behavior of how AC handles the various applications detected in the network.

Using the Application Rule Set on our IP policies

Now we have created two Application Rule Sets, but in their current state they are not doing anything. Application Rule Sets are a template that can be used with our IP policies.

As an example, we open the IP policy used on the Wi-Fi interface related to HTTP. On that policy there is an Application Control tab as shown below.



Enable Application Control: ON

Mode: Use Rule Set

Application Rule Set: Deny_Youtube_P

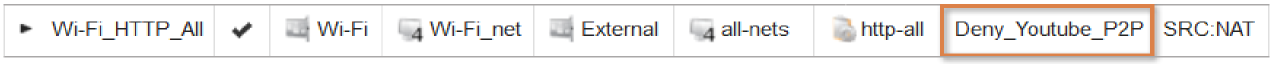
Figure 3.12.12 Activating Application Control on an IP policy

The first setting (**1**) controls if we want to activate AC on this policy or not. Once that option is selected, it will expand to additional options such as the **Mode** (**2**) and Application Rule Set (**3**). Since we have already created an Application Rule Set we can select the one we want to use on the Application Rule Set selection.

Note

If we so choose, we can create an Application Control filter directly on this IP policy. The disadvantage of this is it can only be used on this particular policy. If we want to re-use this application rule filter set we have to re-create it as an Application Rule Set profile.

If we now look at the policy summary for this particular policy it will look like the image below.



► Wi-Fi_HTTP_All	✔	Wi-Fi	Wi-Fi_net	External	all-nets	http-all	Deny_Youtube_P2P	SRC:NAT

Figure 3.12.13 Policy summary when activating AC on an IP policy

As we can see there is now information in the Application field indicating that Application Control is now activated on this particular policy.

From the previous recipe, we configured the Web Policy on this policy where we restricted the webpage access, now we have activated AC on it as well. So not only do we restrict which webpage category that is allowed, we specifically block some applications as well. In our example, we block P2P network access and the ability to connect to Youtube.

Application Control best practice

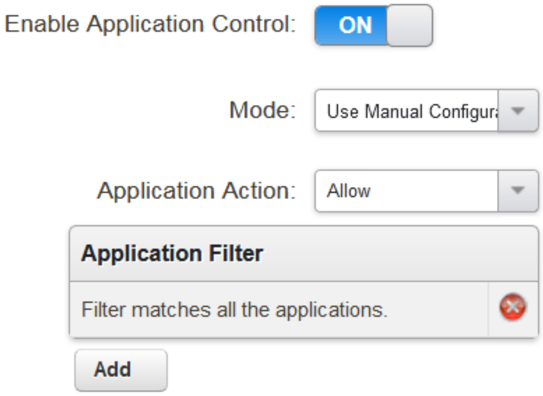
Application Control is a very powerful tool to restrict users from using a particular application, function or even some webpages (such as social media).

But it can also cause problems if we are not careful. Here are a few tips on things to keep in mind when configuring Application Control for the first time.

Step 1: Do not block anything

This can seem a bit strange. Unless we specifically know exactly what kind of applications that exist in the network and which ones we want to allow or deny, it is a good idea to first set up Application Control in such a way that it only logs what is being used in the network.

To do this we either create an Application Rule Set or configure it directly on the relevant IP policies. We will be using a **Default Action** of **Allow**. For the signature filter, we do not select anything particular. Instead, we just open the filter options and press **Add**. This will have the effect that all applications in the database will be used for this policy. This is show in the screenshot below.



Enable Application Control: ON

Mode: Use Manual Configur

Application Action: Allow

Application Filter

Filter matches all the applications.

Figure 3.12.14 Using all available signatures on an AC rule

Since we do not know exactly what may be going on in the network, we first just want to allow all traffic in order to see what is being used and if it should be allowed or not.

Note

Using all applications will put a lot of stress on cOS Core due to the sheer amount of signatures it needs to match for all the traffic. It might be a good idea to restrict this data gathering to one specific network at a time to avoid overloading cOS Core.

Step 2: Make a list of applications that we want to block

Based on the logs, we should be able to get a fairly good view on what kind of applications are being used in the network. Based on that, we make a list of applications that we feel should not be allowed in the network.

Step 3: Make a list of applications that we want to allow

As in the previous step, make a list of applications that we would like to allow in the network.

Compare Notes

Depending on the results we have some decisions to make based on the following questions:

- Do we have a large amount of applications that we want to block?
- Or do we have a large amount of applications that we want to allow?

Based on the answer to the above questions, it should be decided if we want to use Application Rule Sets that use a **Default Action** of **Allow** or **Deny**.

If we only want to allow a very limited amount of applications it is better to use the **Default Action** of **Deny** to avoid having to configure a huge list of applications that should be denied.

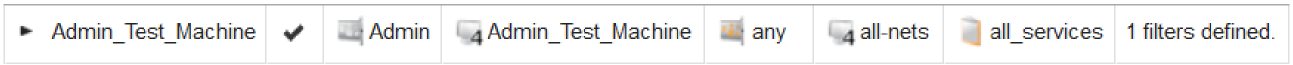
If we want to deny only a few applications we do things the other way, we set the **Default Action** to **Allow** and only configure the specific applications we want to deny.

Either method will work of course, but it will be much less work for the administrator to configure depending on the situation. By using the action **Deny**, we also make sure that any new or unidentified application in the network will be automatically dropped. However, when using **Allow**, the administrator must add a specific **Deny** for the application.

That works the other way around of course. By using allow we would automatically allow any new application in the network. It comes down to preference and the requirements set by the administrator and/or company policy.

Step 5: Set up evaluation policies

Once the desired method has been chosen, it would be a good idea to set up an evaluation policy to verify that the configured Application Rules and filter does what we want. An easy way to do that is to make an IP policy that specifically triggers for one host only, such as the one shown in the next screenshot.



► Admin_Test_Machine	✓	Admin	Admin_Test_Machine	any	all-nets	all_services	1 filters defined.

Figure 3.12.15 An IP policy that only triggers for one source IP

In the above policy, the address object called "Admin_Test_Machine" single IPv4 address (192.168.100.15). So this particular policy will only trigger for one specific host. When we test this, it will not interfere with any other computer behind the Admin interface and is perfect for testing and evaluation.

It is also important that this particular test policy is placed at the top of the policy section related to (in our case) the admin network. The reason for this is we want this policy to trigger first, and we do not want any other policy to interfere with our tests. If, for instance, an HTTP policy was placed above this policy, it would not give us the desired test results for applications that use HTTP as that particular port/protocol would already have triggered another policy.

Step 6: Configure AC on the correct ports and/or protocol to optimize

Another important thing to keep in mind is that applications use specific ports. Some applications may use dynamic ports but, if we take HTTP and FTP for example, they are usually configured on TCP port 80 and 21.

When selecting the applications to use, it is a good idea to therefore verify that the applications selected are actually using the ports/protocols we have specified in our IP policy/policies.

An example would be if we have activated AC on our FTP policy then chosen to block Facebook. That will never work as Facebook will never be accessed on FTP port 21. Avoid using signatures on ports that we know will never trigger, it will put a needless load on cOS Core as it must evaluate signatures that will never trigger.

Step 7: Final implementation and activation

Once we are satisfied that our test host can only access/use the applications we want, and that there is no leakage, we can continue to roll out the changes to our users.

By "leakage", we mean mistakes in the configuration that could cause applications that should be blocked to be allowed and the other way around.

Additional information 1: Overlapping security

Right now, we have activated quite a few security mechanisms and functions. Some of these functions may be doing either a similar or almost the same thing. A good example here is if we go back and first look at *Figure 3.12.9*.

One of the signatures we chose to block here is Youtube. But Youtube also belongs to two Web Content Filtering categories:

- **7** - Entertainment.
- **23** - Music Downloads.

And the Web Content Filtering policy we created earlier for the Wi-Fi network, specifically denies the "Music Downloads" category.

So for our Wi-Fi interface, it would not be needed to specifically block Youtube as that is already disallowed by Web Content Filtering.

It is not a disaster to use both AC and WCF to block Youtube, it is more of an example of how functions can in some situations overlap each other. With today's powerful hardware and processing power, the extra workload caused by such redundancy is in most cases minimal, but it should be kept in mind during configuration so the system can be optimized to work faster.

Additional information 2: Signature Inheritance

The application control signatures have a hierarchical structure and it is important to remember that permissions are also inherited. An example of this is the *http* signature. If the administrator configures application control to block all http traffic they are also blocking all applications that use http such as Facebook. However, if the administrator configures application control to allow the http signature, they are also allowing all applications that use http. For instance, the signature for Facebook is a child of the http signature so allowing http traffic also allows Facebook traffic. If Facebook is to be blocked while still allowing http, it must be blocked separately.

Additional information 3: What if application X Y or Z is not in the database?

Every attempt is made to keep the application control signature database as up to date as possible. However, there are thousands of applications in existence and some are not well known. There can therefore be situations where cOS Core application control cannot properly identify a specific application. If that is the case, Clavister may need help from customers to become aware of obscure, but potentially important applications, so contact the company with details.

Recipe 3.13. Blocking using Schedules

Objectives

Consider the following scenario: During normal school/office hours we want to deny users/students access to Youtube, but after school/office hours they should be allowed to access whatever they want.

The objective of this recipe is to configure schedules to solve this scenario since we want our students to study and pay attention in class instead of looking at videos.

Figure 3.13.1 Blocking Youtube during business hours

Note

This is of course only an example. What to allow or what to deny is up to school or company or the administrator to decide.

There is no general recommendation here and each installation is unique with its own set of requirements.

Detailed Discussion

To accomplish this we will be using a feature called Schedules. The location in the WebUI for configuring schedules is shown in the following screenshot.

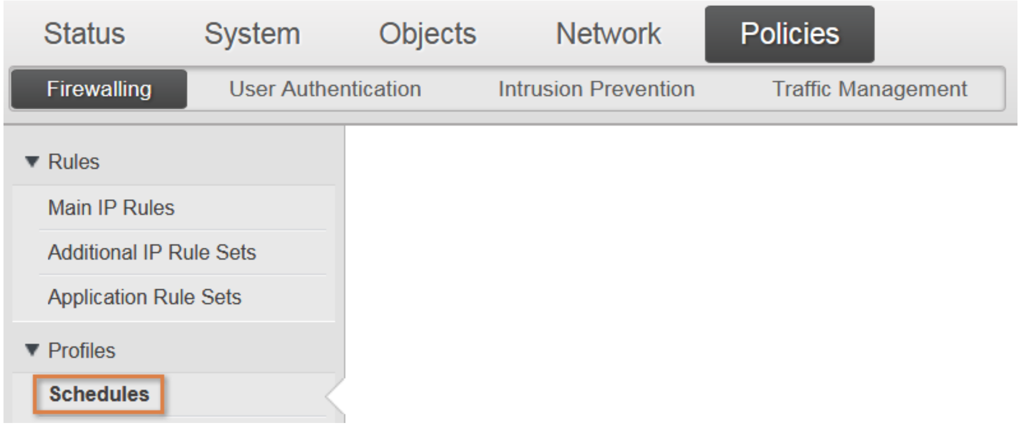


Figure 3.13.2 The WebUI location of schedule profiles

Note

This is not the only location where schedules can be configured. We will also discuss using schedules with traffic shaping in the next chapter.

When creating a schedule profile, there are two types of profile that we can create, as shown in the WebUI screenshot below.

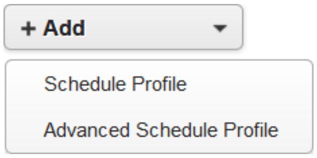


Figure 3.13.3 Creating schedule profiles

The normal **Schedule Profile** allows a schedule with the following specifications to be configured:

- *Only trigger between 07:00:00 to 16:00:00 every day, Monday to Friday.*
- **Or:** *Only trigger between 07:00:00 to 16:00:00 every day, Monday to Friday, between the dates January 1st, 2015 and June 1st, 2015.*

An **Advanced Schedule Profile** allows more advanced profiling to be configured, such as the following requirements:

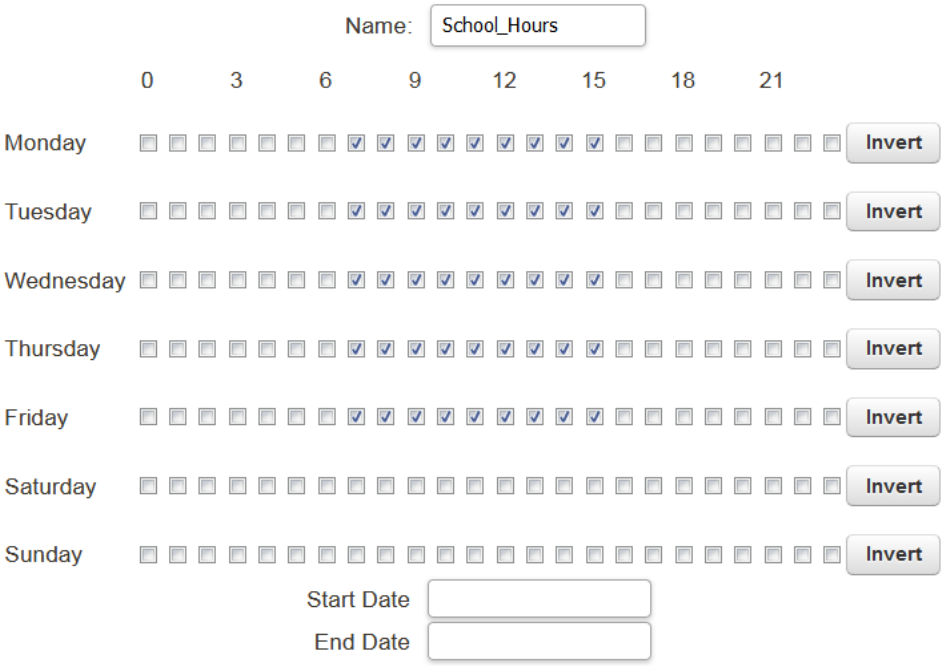
- *Trigger between 07:00:00 and 16:00:00 on the first each month.*
- *Trigger between 07:00:00 and 16:00:00 on the 5th each month.*
- *Trigger between 15:00:00 to 16:00:00 on the each Sunday each week.*

An example scenario

Assume we have the following requirement:

Between 07:00:00 and 16:00:00 each week from Monday to Friday, users connected to the Wi-Fi network should not be able to use Youtube or P2P networks.

The above requirement can be covered using the standard scheduling profile. We create a profile that looks like the one shown in figure 3.13.4.



Name: School_Hours

	0			3			6			9			12			15			18			21			
Monday	□	□	□	□	□	□	□	☑	☑	☑	☑	☑	☑	☑	☑	☑	□	□	□	□	□	□	□	□	Invert
Tuesday	□	□	□	□	□	□	□	☑	☑	☑	☑	☑	☑	☑	☑	☑	□	□	□	□	□	□	□	□	Invert
Wednesday	□	□	□	□	□	□	□	☑	☑	☑	☑	☑	☑	☑	☑	☑	□	□	□	□	□	□	□	□	Invert
Thursday	□	□	□	□	□	□	□	☑	☑	☑	☑	☑	☑	☑	☑	☑	□	□	□	□	□	□	□	□	Invert
Friday	□	□	□	□	□	□	□	☑	☑	☑	☑	☑	☑	☑	☑	☑	□	□	□	□	□	□	□	□	Invert
Saturday	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	Invert
Sunday	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	Invert

Start Date

End Date

Figure 3.13.4 An example standard schedule profile

We intentionally left the "start" and "end" date blank as we want this to be a continuous schedule that never expires.

Note

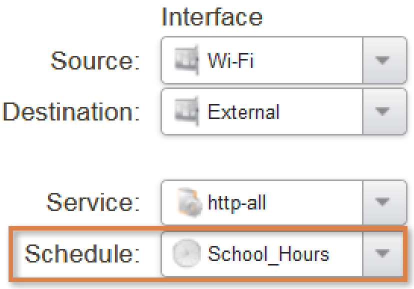
You may notice that we have checked the boxes for 7-15 even though we want the schedule to work between 07:00:00 and 16:00:00 . The reason for this is that if we for example check the box for 15 it means that the schedule will be active between 15:00:00 and 15:59:59. This means that if we would have checked the boxes from 7-16 the schedule would have been active from 07:00:00 to 16:59:59.

Using schedules in policies

Now that we have created the schedule, it is time to apply it to our rule set. In *Recipe 3.12. Using Application Control*, we ended up with a Wi-Fi IP policy where we specifically blocked Youtube and P2P applications.

However, this policy does not have any schedules configured, meaning that this policy will always be active and block the selected applications. Outside school hours, we do not want to enforce this restriction and students should be able to do pretty much what they want.

So we can open our IP policy and apply our newly created schedule profile on the policy, as shown below.



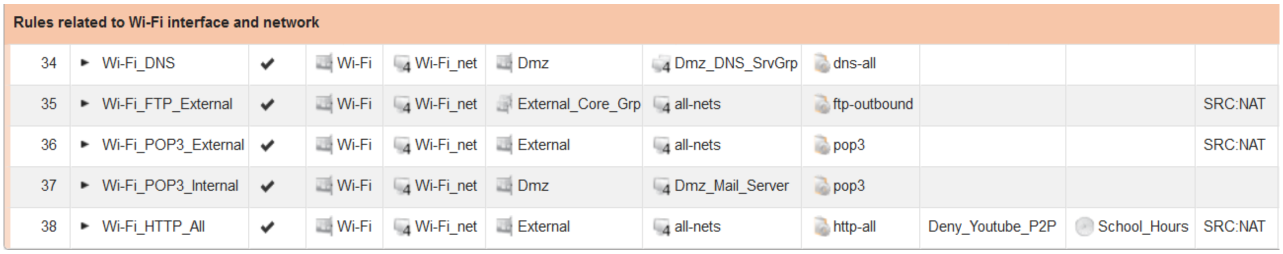
	Interface	Network
Source:	Wi-Fi	Wi-Fi_net
Destination:	External	all-nets
Service:	http-all	
Schedule:	School_Hours	

Figure 3.13.5 Applying a schedule to our previously created Wi-Fi IP policy

This means that this particular policy will only trigger between 07:00 and 16:00 from Monday to Friday.

Review the IP policy before applying the new schedule

Before deploying these changes, let's review the status of the IP policies for the Wi-Fi interface. Our current rule set now looks like the screenshot shown below.



Rules related to Wi-Fi interface and network

34	▸ Wi-Fi_DNS	✔	Wi-Fi	Wi-Fi_net	Dmz	Dmz_DNS_SrvGrp	dns-all			
35	▸ Wi-Fi_FTP_External	✔	Wi-Fi	Wi-Fi_net	External_Core_Grp	all-nets	ftp-outbound			SRC:NAT
36	▸ Wi-Fi_POP3_External	✔	Wi-Fi	Wi-Fi_net	External	all-nets	pop3			SRC:NAT
37	▸ Wi-Fi_POP3_Internal	✔	Wi-Fi	Wi-Fi_net	Dmz	Dmz_Mail_Server	pop3			
38	▸ Wi-Fi_HTTP_All	✔	Wi-Fi	Wi-Fi_net	External	all-nets	http-all	Deny_Youtube_P2P	School_Hours	SRC:NAT

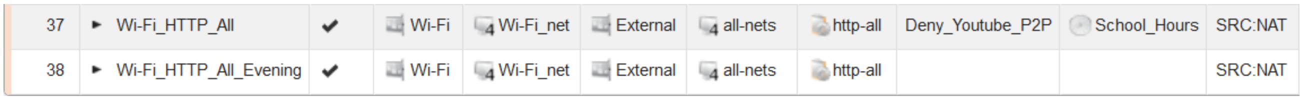
Figure 3.13.6 IP policies with schedules applied

We can see that our schedule for "School_Hours" is in place on the IP policy we created earlier. This means that our scheduling policy will trigger between 07:00 and 16:00 and block the categories chosen by Web Content Filtering and Application Control.

But what happens before 07:00 and after 16:00? Which policy will trigger then? The answer is: **No policy will trigger**. All HTTP/HTTPS traffic will be blocked during these times. On weekends it will be blocked as well as the schedule IP policy only triggers between Monday and Fridays. To solve this problem we must create another IP policy that should trigger for all the periods that are not covered by our schedule policy.

The question then arises, should we be less restrictive with our Wi-Fi network outside school hours?

That sounds reasonable, so we create a new Web Profile that is less restrictive and only block some of the major undesired categories. We then create an IP policy that uses that Web Profile and http-all service, as shown below. We will not activate application control on this policy. (restrictions will naturally vary depending on the requirements)



37	► Wi-Fi_HTTP_All	✓	Wi-Fi	Wi-Fi_net	External	all-nets	http-all	Deny_Youtube_P2P	School_Hours	SRC:NAT
38	► Wi-Fi_HTTP_All_Evening	✓	Wi-Fi	Wi-Fi_net	External	all-nets	http-all			SRC:NAT

Figure 3.13.7 Creating additional IP policies with schedules

It is very important that the policy is placed in the correct order. Since we want our schedule policy to trigger during its configured time intervals, it MUST be placed above our evening IP policy.

Only when our schedule policy does NOT trigger, will it fall down to trigger on our general policy (#38). As mentioned previously, IP policies are always read and interpreted from the top to the bottom of the rule set.

Recipe 3.14. Using VLANs to build a Lab network

Objectives

So far in this chapter, we have not mentioned the lab network very much. That is because it is quite different from our other interfaces.

The idea of the lab network is that on this particular network, teachers and students should be able to experiment with all kinds of network equipment, programs and prototypes, without restrictions.

When doing work on something like an application or network related project, the student/teacher would not want interference from some of the security systems. If they encounter a problem, how would they know if it's related to their lab or some external security system?

Having such unrestricted access is also very dangerous. Such networks must be kept isolated and closed off from all other networks as much as possible.

The objective of this recipe is to discuss how we can implement a lab network that does not interfere with other interfaces and our network security.

Detailed Discussion

In our current configuration for the LAB interface, we only have the network 192.168.0.0/24 routed on it. Having all students and teachers using the same lab network would be very chaotic as it will be flooded with packets from everyone's machines. This can cause interference from each other and will be a generally bad lab environment.

So how do we solve this? We do not have any more physical interfaces on the Clavister firewall, but we can solve the problem by using a Virtual LAN (VLAN)!

VLAN explained

Virtual LAN (VLAN) support in cOS Core allows the definition of one or more Virtual LAN interfaces which are associated with a particular physical interface. These are then considered to be logical interfaces by cOS Core and can be treated like any other interfaces in cOS Core IP rule sets and routing tables.

VLANs are useful in several different scenarios. A typical application is to allow one Ethernet interface to appear as many separate interfaces.

This means that the number of physical Ethernet interfaces on a Clavister Next Generation Firewall need not limit how many totally separated external networks can be connected.

Another common use of VLANs is to group together clients in an organization so that the traffic belonging to different groups is kept completely separate in different VLANs.

Traffic can then only flow between the different VLANs under the control of cOS Core and is filtered using the security policies described by the cOS Core rule sets.

Segmenting the lab network using VLANs

In our lab network example we will have 5 different lab networks (6 if we count the physical LAB interface/network itself) each given their own VLAN and VLAN ID as illustrated below in *Figure 3.14.1*.

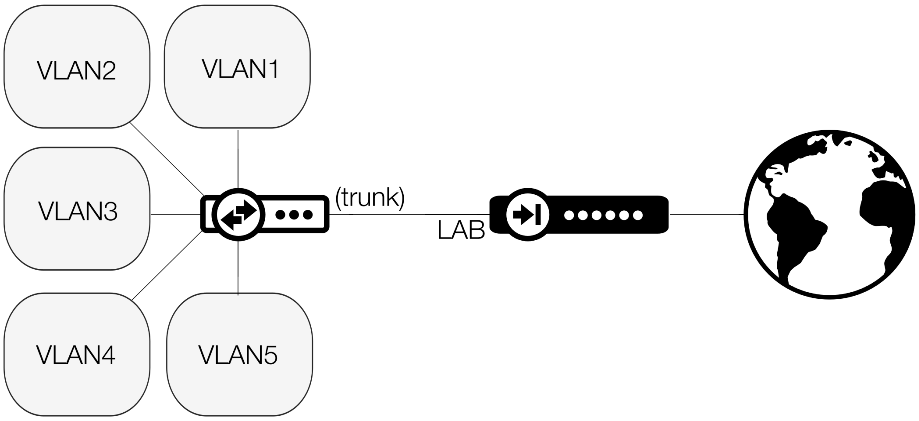


Figure 3.14.1 Segmenting the lab network using VLANs

As listed below in *Table 3.14.2*, we will allocate network(s) to the various VLAN interfaces as well as the physical trunk LAB interface (a trunk interface is basically an interface multiple VLANs are traversing).

Interface Name	VLAN ID	Network
Lab	Base interface/trunk	192.168.0.0/24
Lab_VLAN_01	1	192.168.1.0/24
Lab_VLAN_02	2	192.168.2.0/24
Lab_VLAN_03	3	192.168.3.0/24
Lab_VLAN_04	4	192.168.4.0/24
Lab_VLAN_05	5	192.168.5.0/24

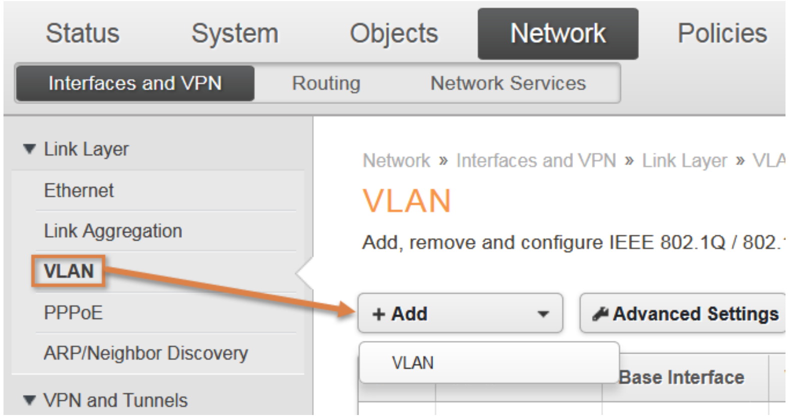
Table 3.14.2 Overview of the LAB network's VLAN IDs and network for each segment

Before we create the actual VLAN interfaces, we go to the address book and create all the objects needed for the new interfaces. We need two objects for each interface, an IP address and a network.

For the IP, we will choose the first address in each network (192.168.1.1, 192.168.2.1, 192.168.3.1 and so on). This IP will be used as default gateway for all the lab machines attached to the corresponding network segment.

Creating VLAN interfaces

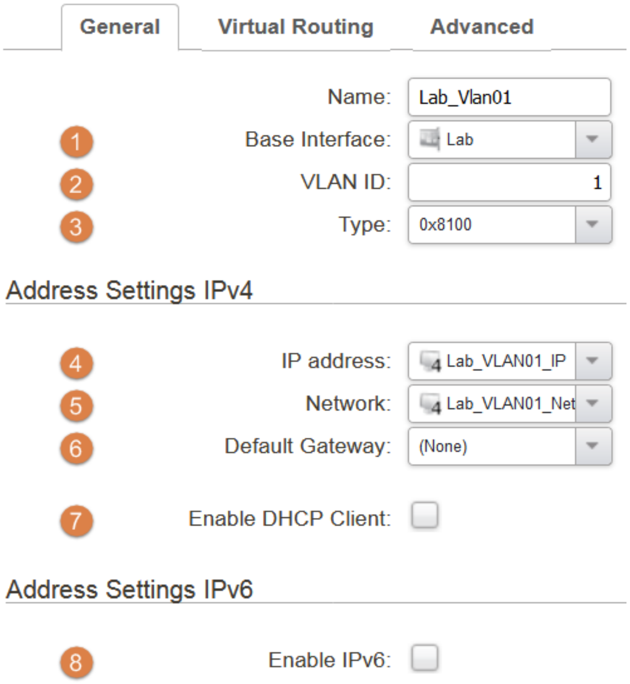
To create a VLAN, we go to the WebUI location shown in the screenshot below and select **Add** then **VLAN**.



Add, remove and configure IEEE 802.1Q / 802.

Figure 3.14.3 Creating a VLAN

There are a large number of options for a VLAN interface and these are shown in the screenshot below. We will go into detail about what each option does.



Name: Lab_Vlan01

1 Base Interface: Lab

2 VLAN ID: 1

3 Type: 0x8100

Address Settings IPv4

4 IP address: Lab_VLAN01_IP

5 Network: Lab_VLAN01_Net

6 Default Gateway: (None)

7 Enable DHCP Client: □

Address Settings IPv6

8 Enable IPv6: □

Figure 3.14.4 VLAN options

The **Base interface** (**1**) determines on which interface the VLAN should be attached. When traffic is being sent or received on this particular VLAN, the physical interface LAB will be used.

The **VLAN ID** (**2**) determines which VLAN ID we should give this interface. In this case we will use a VLAN ID of **1** as that matches the network we allocate to this VLAN (192.168.1.0/24).

The **Type** (**3**) contains the option to decide what kind of VLAN we want this to be. Without going into too much detail, there are primary two types: Normal VLAN and Service (QinQ) VLAN.

Normal VLAN and Service VLAN

When configuring normal VLANs, there is a limit of 4096 VLANs per physical interface. In some really large networks, this can be not enough. In that case we can use Service VLANs which means we can have a VLAN inside another VLAN, dramatically increasing the number of VLANs that can be configured and used in the network. For now, we will leave it as the default type (0x8100) as we do not have the need for a large amount of VLANs at this time. However, it is a University so there is a fairly good chance it could be needed in the future as the networks grow.

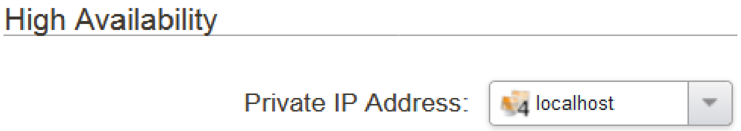
For the **IP address** (**4**) and **Network** (**5**) options, we use the IP objects we created earlier from the address book. The names are a little long but don't be afraid to use long names in policies and object names. A good descriptive object name can be extremely useful.

We will not be using a **Default Gateway** (**6**) because the lab computers will be directly connected to a switch and not go through any routers (other than cOS Core).

The **DHCP client** option (**7**) will not be used either. As we use a static IP address on all the VLANs.

We are not using any IPv6 at this stage so we will leave the **Enable IPv6** option (**8**) disabled.

The last option we need to configure on each VLAN is the HA IP address. Since this is a High Availability cluster we must define the private IP address the cluster should use on these VLANs, it is a required field when HA is enabled. This option is shown in the screenshot below.



High Availability

Private IP Address: localhost

Figure 3.14.5 Setting the HA private IP address for the VLAN interface

Using Localhost as the HA object in VLANs

As we can see in the image above, we have chosen to use the default object called "localhost". This object means that the private IP address will be 127.0.0.1 and 127.0.0.2 on all the VLANs.

The reason why we use this particular object is because there is no direct need to dedicate IP addresses to the private IP address as these IPs are primarily used by cOS Core when cOS Core itself tries to reach something past this interface. An example would be if we have a log receiver configured on this VLAN or if we ping something in this network from the CLI. Such a ping will be initiated from cOS Core itself and it must have a valid IP address to use as the sender in order to succeed.

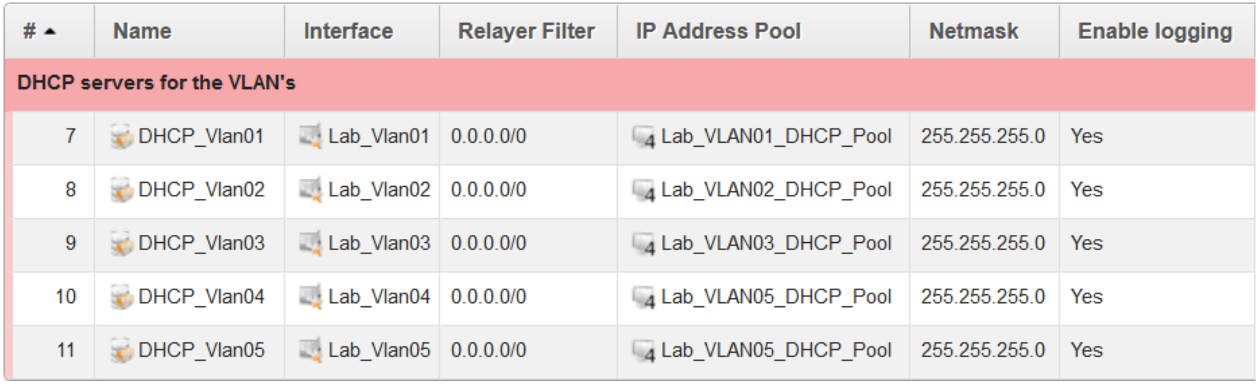
This is not a big problem as we can easily use a sub-parameter for the ping to select which IP we want to use as sender, but we will not go into detail regarding that until later.

A route will be automatically added to the main routing table for these VLANs by default. If we want to structure the routing table, it is recommended to remove the auto-add-route option in order to place the VLAN routes in a comment group for easier visualization of the routing table.

Adding additional DHCP servers to each VLAN segment

Similar to our physical interface, we also want to configure DHCP servers for our VLANs. It will be very cumbersome if there are many lab computers and all have to be configured manually, especially if these user devices move around a bit between the lab networks.

To make things easier, we will add a DHCP server for each VLAN for easier lab management and ease of access for our lab users. This is shown in the DHCP server list screenshot below.



# ▲	Name	Interface	Relayer Filter	IP Address Pool	Netmask	Enable logging
DHCP servers for the VLAN's						
7	DHCP_Vlan01	Lab_Vlan01	0.0.0.0/0	Lab_VLAN01_DHCP_Pool	255.255.255.0	Yes
8	DHCP_Vlan02	Lab_Vlan02	0.0.0.0/0	Lab_VLAN02_DHCP_Pool	255.255.255.0	Yes
9	DHCP_Vlan03	Lab_Vlan03	0.0.0.0/0	Lab_VLAN03_DHCP_Pool	255.255.255.0	Yes
10	DHCP_Vlan04	Lab_Vlan04	0.0.0.0/0	Lab_VLAN05_DHCP_Pool	255.255.255.0	Yes
11	DHCP_Vlan05	Lab_Vlan05	0.0.0.0/0	Lab_VLAN05_DHCP_Pool	255.255.255.0	Yes

Figure 3.14.6 Summary of DHCP servers configured on various VLANs

Each Lab VLAN DHCP Pool will consist of roughly 150 IP addresses.

Configuring IP policies for the lab network

We now come to the most important part, the IP policies. How should we configure the network access for the lab network. Should users on the LAB network be allowed to initiate connections to any other network in the University?

The answer is NO, as illustrated below in *Figure 3.14.7*.

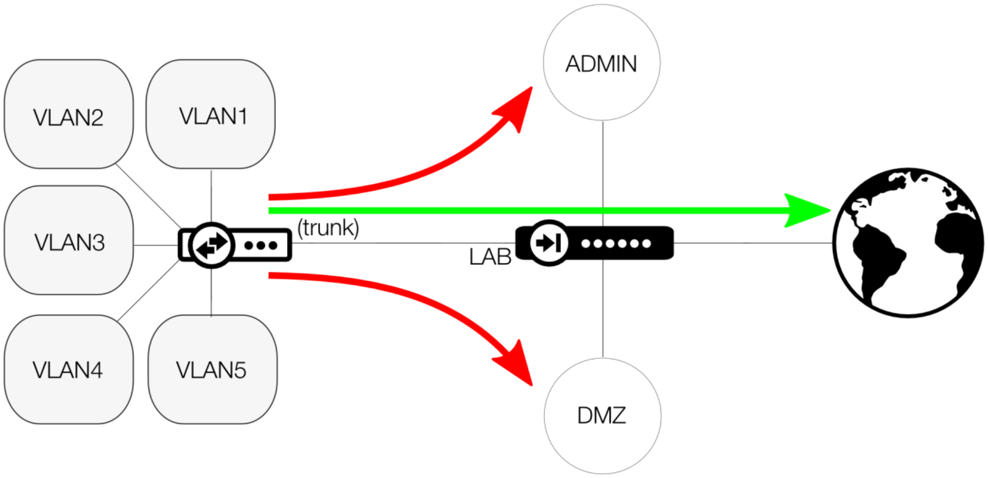


Figure 3.14.7 Restricting Lab network access

As was previously stated, the lab network will be fairly unrestricted. We will not be using any ALGs, Anti-Virus, Web Content Filtering or Application Control on this network. This reason is that doing so may cause interference to lab equipment and the various tests being done on this network.

What about incoming connections to the lab network, should we allow that? The answer is again, NO. We do not want to allow incoming connections directly to the lab network from outside sources either.

This could however cause problems for some labs that may require incoming connections due to one reason or another. Allowing incoming connections will be discussed later. For now, we will not allow it in our IP policies.

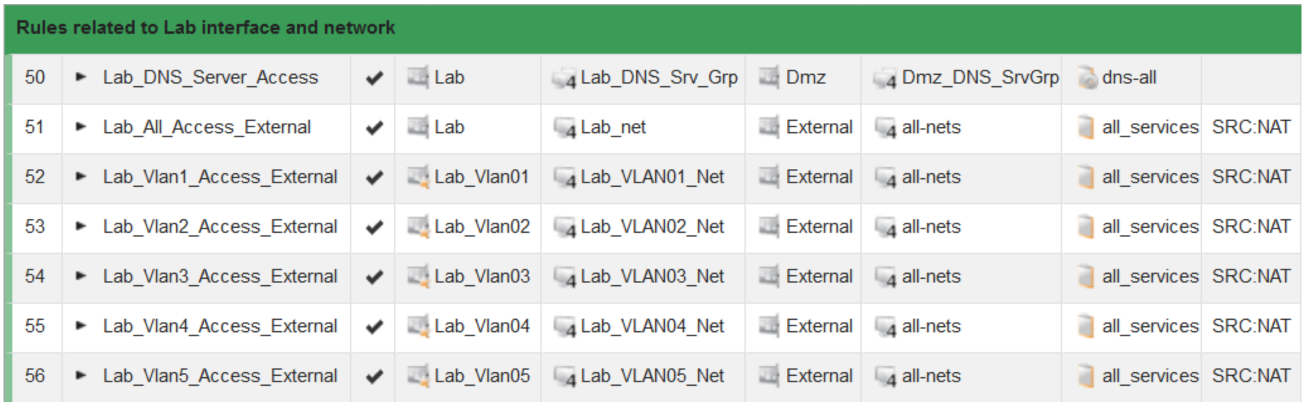
Communication between each lab segment will not be allowed either and we will keep them separated for a reason. We will discuss possible ways to override this temporarily later.

As we will not allow connections from the lab network to, for example, the DMZ and LAB network users will be unable to perform DNS queries. To solve this particular problem we setup two DNS servers behind the physical Lab interface that all users in the lab network will be able to use.

Note

An alternative DNS solution would be to allow lab users to connect to the ISP DNS servers directly. This may not be a desirable way to solve the DNS issue as there is a very high probability that the administrator has internal domain or DNS names that would be impossible to resolve if using external DNS servers.

The screenshot below shows the policies configured for the various lab networks.



Rules related to Lab interface and network

50	► Lab_DNS_Server_Access	✓	Lab	Lab_DNS_Srv_Grp	Dmz	Dmz_DNS_SrvGrp	dns-all	
51	► Lab_All_Access_External	✓	Lab	Lab_net	External	all-nets	all_services	SRC:NAT
52	► Lab_Vlan1_Access_External	✓	Lab_Vlan01	Lab_VLAN01_Net	External	all-nets	all_services	SRC:NAT
53	► Lab_Vlan2_Access_External	✓	Lab_Vlan02	Lab_VLAN02_Net	External	all-nets	all_services	SRC:NAT
54	► Lab_Vlan3_Access_External	✓	Lab_Vlan03	Lab_VLAN03_Net	External	all-nets	all_services	SRC:NAT
55	► Lab_Vlan4_Access_External	✓	Lab_Vlan04	Lab_VLAN04_Net	External	all-nets	all_services	SRC:NAT
56	► Lab_Vlan5_Access_External	✓	Lab_Vlan05	Lab_VLAN05_Net	External	all-nets	all_services	SRC:NAT

Figure 3.14.8 IP policy summary for lab network interfaces

The current IP policy setup

Let's make some comments about the IP policies shown above.

IP policy **50** allows the DNS servers located in the Lab network to communicate with the DNS servers in the DMZ. This is for DNS queries only. No other access is allowed from the Lab to the DMZ network (or any other network/interface other than the external).

IP policies **51** to **56** are all identical. They allow unrestricted access for the various lab interfaces and networks to communicate out onto the Internet. As this is a lab network we have not activated any security mechanisms at all because that could be in conflict with the idea of having a lab/test network without anything interfering with its traffic.

This could be a potential problem but later we will go into more details about how to lock down the LAB networks even further.

Terminal server access to the lab network

As discussed earlier, we do not want to allow any incoming connections to the lab network for machines that are beyond our control. However, what we can do is to setup "thin" clients/terminal servers for each lab VLAN and network. Then we can allow users from their respective interfaces (such as TEACHERS and DORMITORY) to access the lab networks by connecting to these terminal servers.

These terminal servers are set up by the administrator and so are better controlled and protected, lowering the security risks involved. This is illustrated next in *Figure 3.14.9*.

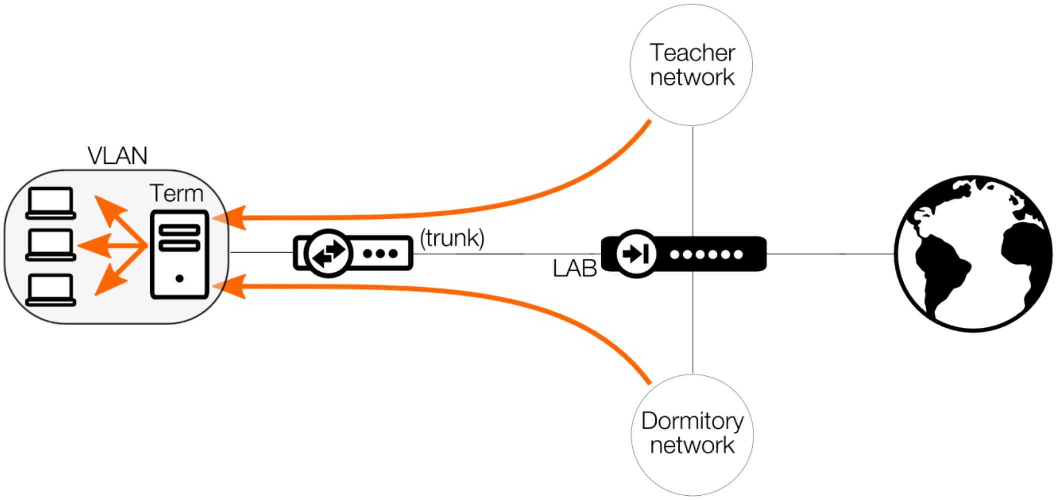


Figure 3.14.9 A terminal server allowing remote access to the lab network

In today's network environment it would be very cumbersome to only allow our students and teachers access to the lab network by having a physical presence on that particular network.

Students must be able to work and access their projects remotely from their DORMITORY network and possibly even clients on the Wi-Fi and TEACHERS interfaces must be able to help the students and check on their progress. Using the thin client solution is an easy way to provide such access without compromising the overall network security.

Since this is a university, one server per lab interface may not be sufficient. As the network grows larger, so will be the need for additional servers and more lab networks and equipment.

Recipe 3.15. Assigning additional interface IPs

Objectives

The objective of this recipe is to learn how to assign multiple IP address to a physical (or VLAN) interface. The most common scenario for this is when we have requested and received multiple public IP address from our ISP and we want to assign them all to the external interface.

The most common reason for this need is when an organization has a multitude of different servers that will use the same destination port. Since we cannot use the same port multiple times towards the same IP we need additional IP addresses so we can reuse the port number. For example, for the HTTP and HTTPS ports, 80 and 443.

Detailed Discussion

As an example, we will use the EXTERNAL interface of our Clavister Next Generation Firewall. Currently, this has only one public IP address assigned to it, which is 203.0.113.10. Now, we want to assign the public IP addresses from 203.0.113.11 to 203.0.113.15 to the EXTERNAL interface, as shown below in *Figure 3.15.1*.

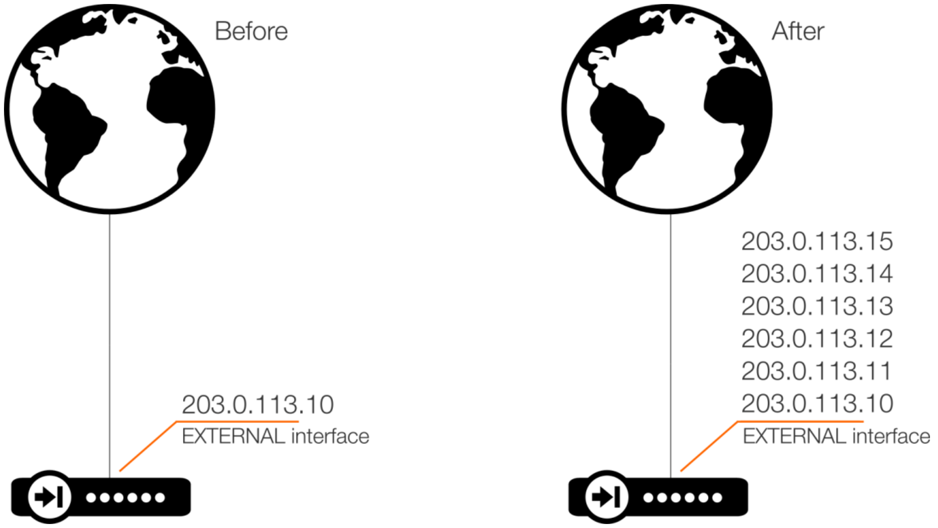
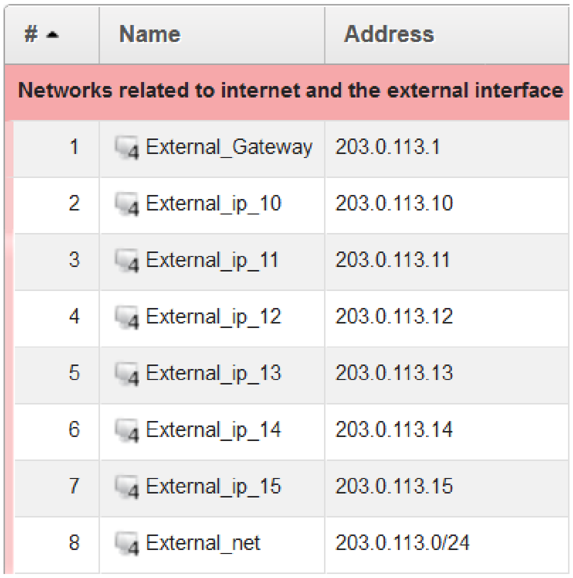


Figure 3.15.1 Assigning multiple IP addresses

Creating additional network objects

In order to keep things orderly, we first create additional network objects for the new IP addresses as shown below.



# ▲	Name	Address
Networks related to internet and the external interface		
1	External_Gateway	203.0.113.1
2	External_ip_10	203.0.113.10
3	External_ip_11	203.0.113.11
4	External_ip_12	203.0.113.12
5	External_ip_13	203.0.113.13
6	External_ip_14	203.0.113.14
7	External_ip_15	203.0.113.15
8	External_net	203.0.113.0/24

Figure 3.15.2 Additional address book objects for IPs

It is very likely that these objects will be used in various IP policies and other functions, so creating and using address book objects is highly recommended.

Two methods of adding extra IP addresses to an interface

There are two ways to add additional IP addresses to an interface. This can be slightly confusing as neither of these methods configures the IPs directly on the interface itself.

Method 1 – ARP Publish

The first method uses a feature called "ARP/Neighbor Discovery". In the WebUI, this feature can be found under **Network > Interfaces > Link Layer** and in figure 3.15.3.

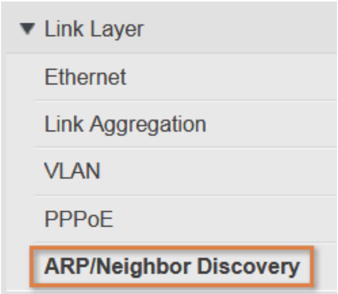
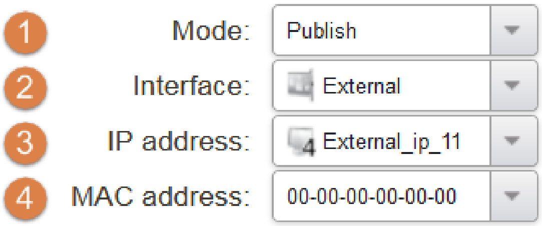


Figure 3.15.3 Using ARP/Neighbor Discovery

Note

We will not go into details about what exactly ARP is at this stage. For more information about ARP please see the end of this recipe.

When creating a new ARP entry we are presented with a couple of options as shown below. Let's look at the various options.



1 Mode: Publish
2 Interface: External
3 IP address: External_ip_11
4 MAC address: 00-00-00-00-00-00

Figure 3.15.4 ARP options

By default, a **Mode** (**1**) called "Publish" is selected. This is the mode chosen 99% of the time. For information about the other modes please see the end of this recipe.

The option "Publish" means that we tell cOS Core that it should respond to ARP queries on the selected interface (**2**) and for the chosen IP address (**3**). This means that when an external source (usually the Internet Service Provider router) asks who owns this IP address, cOS Core will say: "I own this IP address, and my MAC address (**4**) is this."

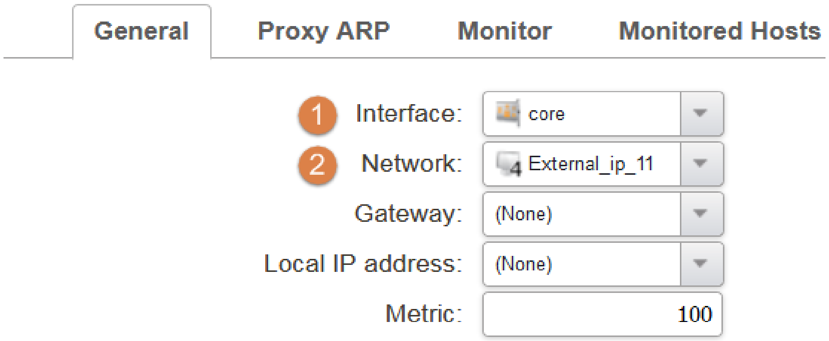
If the **MAC address** field is left at the default value of *00-00-00-00-00-00* (**4**), cOS Core will use the MAC address of the chosen interface (**2**).

Using ARP Publish is the most common choice for having multiple IP address assigned to same interface because having the same MAC address associated with the addresses is not a problem.

Method 2 – Proxy ARP

The second method of adding multiple IP address to a single interface is called "Proxy ARP". This method is configured in the WebUI by going to **Network > Routing**.

In order to create our Proxy ARP entry, we create a new *Core* route, as shown in the next screenshot.



Interface: core
Network: External_ip_11
Gateway: (None)
Local IP address: (None)
Metric: 100

Figure 3.15.5 Adding a core route

As the interface (**1**), we must choose "Core". Next, is the IP address we want to assign to cOS Core, so we select our chosen IP address (**2**). More details about the Core interface can be found in *Recipe 2.5. Configuring a DMZ and allowing access to an internal web server.*

Note

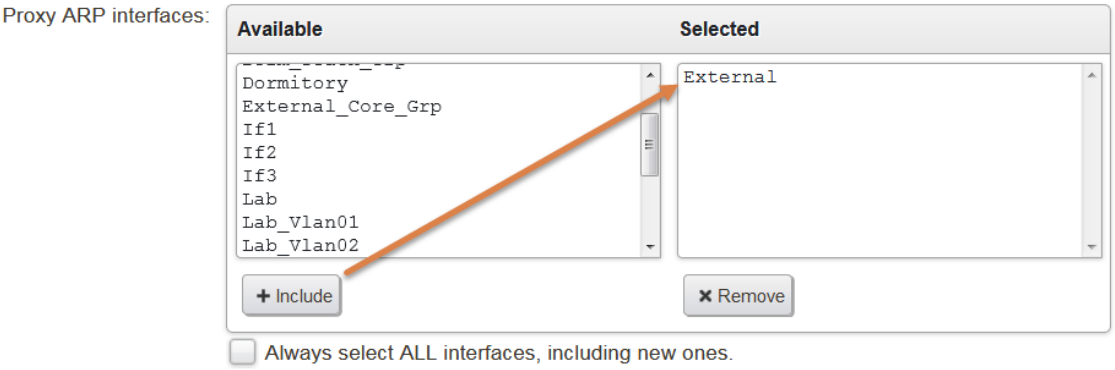
We could of course group of all our IP addresses in a single route instead of one route per IP, but to keep things simple and to the point we create multiple routes in this example. It will also make the routing table easier to read and understand.

Basically, we can interpret the route like this: "*Assign IP address 203.0.113.11 to the Core interface*".

But what happens if we select the external interface instead of Core? If we then read the route, it will be interpreted like this: "*In order to reach 203.0.113.11, use interface External*".

So the behavior will be quite different depending on whether we use the Core interface or not. However, simply selecting the Core interface is not enough. What we have done so far is to tell cOS Core that it has another IP address, but not if it should respond to ARP for this IP address on any interface.

For that, we need to go to the Proxy ARP tab, which is shown in the screenshot below.



Proxy ARP interfaces:

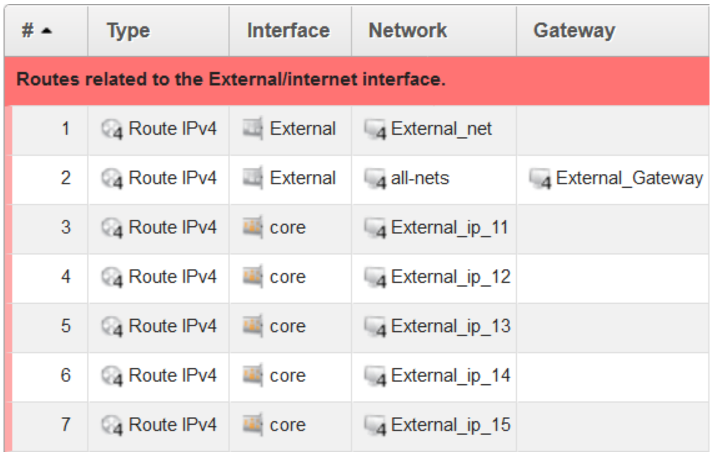
Available	Selected
Dormitory	External
External_Core_Grp	
If1	
If2	
If3	
Lab	
Lab_Vlan01	
Lab_Vlan02	

□ Always select ALL interfaces, including new ones.

Figure 3.15.6 Configuring Proxy ARP

Here, we simply select the interface(s) on which we want cOS Core to respond to ARP. In this example, it is only one interface: EXTERNAL.

Now, we are all set. When ARP queries arrive on the External interface asking about our chosen IP addresses, cOS Core will reply and say that this IP belongs to cOS Core. After we have added all of our new Core routes, the routes for the external interface will now look like the list in figure 3.15.7.



# ▲	Type	Interface	Network	Gateway
Routes related to the External/internet interface.				
1	Route IPv4	External	External_net	
2	Route IPv4	External	all-nets	External_Gateway
3	Route IPv4	core	External_ip_11	
4	Route IPv4	core	External_ip_12	
5	Route IPv4	core	External_ip_13	
6	Route IPv4	core	External_ip_14	
7	Route IPv4	core	External_ip_15	

Figure 3.15.7 External routes

Which method is best to use?

Both methods work if we simply want cOS Core to respond to ARP for an IP address. However, they both have strengths and weaknesses.

ARP Publish advantages: Clear indication of what it does. More customized options regarding MAC address and static ARP possibility.

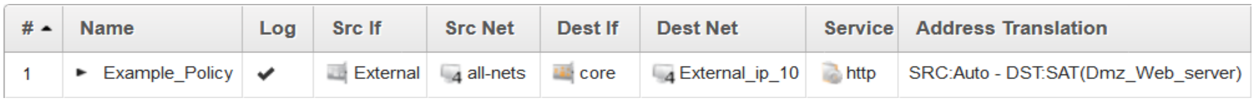
ARP Publish disadvantages: Can only be done on one IP address at a time. Not possible to create Core routes. IP policies needs to be configured differently than for an interface IP address.

Proxy ARP advantages: Can be done on an entire network. Can select multiple interfaces at once. IP policy configuration will be the same as for a physical interface.

Proxy ARP disadvantages: Not possible to use XPublish. Not possible to create static ARP entries. Difficult to get an overview.

Differences in policy creation between ARP publish and Proxy ARP

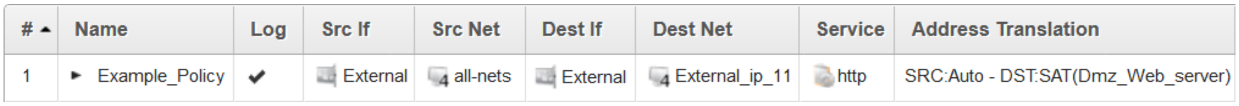
Which method we use will also have an impact on our how IP policies need to be configured. If we start by using a physical interface IP address for an incoming destination SAT policy, as an example, it may look like the rule set below.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
1	► Example_Policy	✓	External	all-nets	core	External_ip_10	http	SRC:Auto - DST:SAT(Dmz_Web_server)

Figure 3.15.8 Example of interface destination SAT policy

Now, if we apply the same policy setup but with an ARP published IP using method one, it will look like the following rule set.

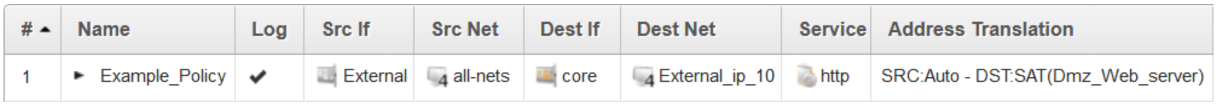


# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
1	► Example_Policy	✓	External	all-nets	External	External_ip_11	http	SRC:Auto - DST:SAT(Dmz_Web_server)

Figure 3.15.9 Destination SAT policy using an ARP published IP as destination

As we can see, the destination interface is no longer Core but rather the External interface. It can be a little confusing as cOS Core will still reply to incoming ARP queries towards this IP. Basically, ARP Publish is a legacy cOS Core feature that has existed for a long time. The need to be able to ARP publish entire networks led to the introduction of Proxy ARP into cOS Core. It was also made possible to Proxy ARP IP address on the Core interface itself to make it behave more like an interface IP address.

If we apply the above policy example on a Core routed Proxy ARPed IP address, it would look like the following.



# ▲	Name	Log	Src If	Src Net	Dest If	Dest Net	Service	Address Translation
1	► Example_Policy	✓	External	all-nets	core	External_ip_10	http	SRC:Auto - DST:SAT(Dmz_Web_server)

Figure 3.15.10 Destination SAT policy using a Core routed Proxy ARPed IP

As we can see, there is no difference between the above and the original destination SAT policy. This is one of the advantages of using Core routes and Proxy ARP. When creating the IP policies they are configured in the same as for any other interface.

Optional: How to find out if an IP is Core routed or not

Sometimes, it can be confusing to know how the policies should be defined when it comes to the source or destination interface. Using "Any" as the interface is, of course, one solution but the more we restrict the policies to use specific interfaces, the better we lock down the network and the more secure it will be.

Having policies that can trigger on multiple interfaces and/or networks could allow our network security to be breached.

An easy way to find out where an IP address is routed is to use the cOS Core CLI. Connect to the CLI and use the following command:

```
Device:/> routes -lookup=<ip>
```

The output from this command might look something like the following:

```
Device:/> routes -lookup=8.8.8.8
Looking up 8.8.8.8 in routing table "main":

Matching route: 0.0.0.0/0
Routing table : main
Send via iface: EXTERNAL
Gateway       : 203.0.113.1

Proxy ARP on  :
Local IP      : (use iface IP in ARP queries)
Metric        : 100
Flags         :
```

If we look at the "Send via iface" value in the CLI example, it is the "EXTERNAL" interface so this is the interface we need to use in our policy.

Optional: Information about ARP

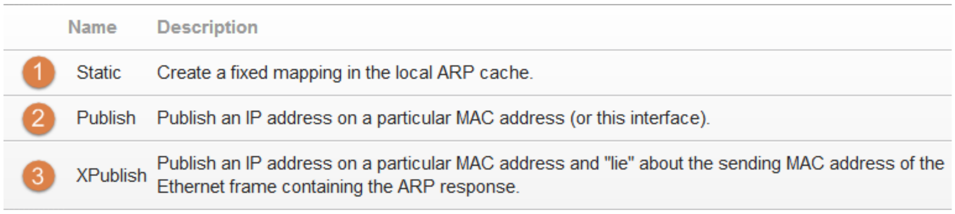
Address Resolution Protocol (ARP) allows the mapping of a network layer protocol (OSI layer 3) address to a data link layer hardware address (OSI layer 2). In data networks it is used to resolve an IPv4 address into its corresponding Ethernet address. ARP operates at the OSI layer 2, data link layer, 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 IPv4 address to get the corresponding Ethernet address, it broadcasts an ARP request packet. The ARP request packet contains the source MAC address, the source IPv4 address and the destination IPv4 address. Each host in the local network receives this packet. The host with the specified destination address sends an ARP reply packet to the originating host with its MAC address.

Optional: Additional ARP modes when using ARP publish

When describing method one, ARP Publish, we used Publish mode but there are some other settings here that could be of interest depending on the scenario. These modes are shown in figure 3.15.11.



	Name	Description
1	Static	Create a fixed mapping in the local ARP cache.
2	Publish	Publish an IP address on a particular MAC address (or this interface).
3	XPublish	Publish an IP address on a particular MAC address and "lie" about the sending MAC address of the Ethernet frame containing the ARP response.

Figure 3.15.11 The different ARP modes

Static Mode

A Static ARP object inserts a mapping into the cOS Core ARP cache which connects a specified IP address with the associated Ethernet interface's MAC address. This mode is not for publishing the address for external devices but rather for telling cOS Core itself how to reach external devices.

A static ARP entry tells cOS Core that a specific IP address can be reached through a specific interface using a specific MAC address. This means, that when cOS Core wants to communicate with the address, it consults the ARP table static entries and it can determine that it can be reached at a specific MAC address on a specific interface.

The most frequent use of static ARP objects is in situations where some external network device is not responding to ARP requests correctly and is reporting an incorrect MAC address. Some network devices, such as wireless modems, can have these problems. It may also be used to lock an IP address to a specific MAC address for increasing security or to avoid denial-of-service attacks if there are rogue users in a network.

However, such protection only applies to packets being sent to that IP address. It does not apply to packets being sent from that IP address.

Publish and XPublish Modes

With Publish and XPublish modes, the ARP object creates an association between an IP address and a MAC address for publishing on the interface to external devices. If the MAC address is not specified, the MAC address of the associated Ethernet interface is used.

To understand the difference between Publish and XPublish, it is necessary to understand that when cOS Core responds to an ARP query, there are two MAC addresses in the Ethernet frame sent back with the ARP response. These two addresses are shown next in *Figure 3.15.12*.

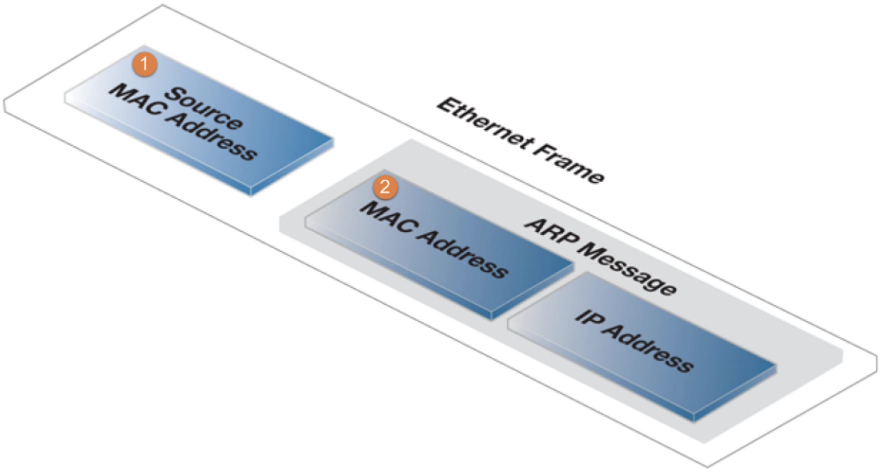


Figure 3.15.12 The Ethernet frame of an ARP response

The two MAC addresses illustrated in *Figure 3.15.12* are the following:

1. The MAC address in the Ethernet frame of the Ethernet interface sending the response.
2. The MAC address in the ARP response which is contained within this frame. This is usually the same as **1**, the source MAC address in the Ethernet frame, but does not have to be.

The Publish option uses the real MAC address of the sending interface for the address (**1**) in the Ethernet frame.

In rare cases, some network equipment will require that both MAC addresses in the response (**1** and **2**) are the same. In this case, XPublish is used since it changes both MAC addresses in the response to be the published MAC address. In other words, XPublish "lies" about the source address of the ARP response.

If a published MAC address is the same as the MAC address of the physical interface, it will make no difference if Publish or XPublish is selected, the result will be the same.

Recipe 3.16. Assigning Public IPs to protected hosts

Objectives

The objective of this recipe is to assign public IP addresses to several computers located in the University LAB network. We will use routing to tell cOS Core where to find the public IP addresses and how cOS Core should behave when the host wants to communicate with the outside world.

Detailed Discussion

This recipe will go into details about network and ARP default behaviors on an advanced level. It's therefore recommended that the user has read and understood *Recipe 3.15. Assigning additional interface IPs* before proceeding.

A very common scenario is when the administrator would like to assign a public IP address directly to a machine behind the Clavister firewall. In our example scenario, we want to assign one public IP address to each host computer in our lab network. This makes five IP addresses in total, as shown below in *Figure 3.16.1*.

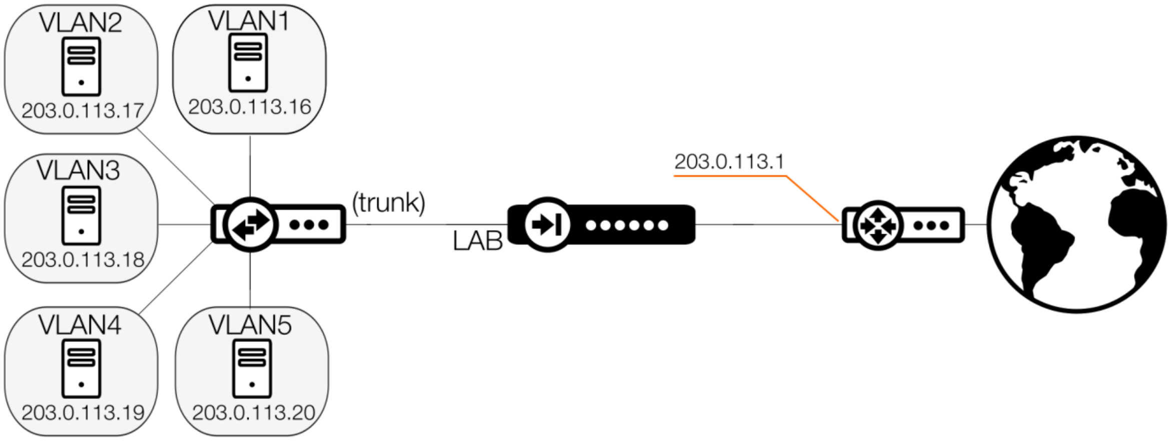
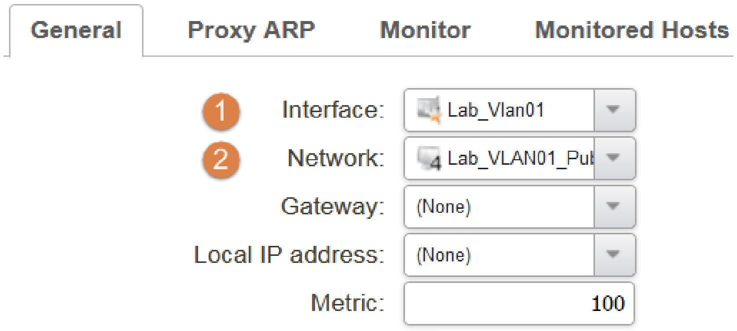


Figure 3.16.1 Assigning public IPs to LAB VLANs

We solve this using routing because we do not want to assign these IP addresses to a cOS Core interface as we did previously.

We go to the routing table *main* and create a new route with the parameters shown in the next screenshot.



Interface: Lab_Vlan01

Network: Lab_VLAN01_Pul

Gateway: (None)

Local IP address: (None)

Metric: 100

Figure 3.16.2 A route for a public IP towards a LAB VLAN

As the route's interface (**1**), we choose the interface the target host is located behind. Prior to this route creation, we will have already created an object in the address book for one of the public IP addresses and we will use this as our network (**2**) on our route. In this example, the object created corresponds to the IP address 203.0.113.16.

This newly created route means: "*In order to find 203.0.113.16, send out an ARP request directly on the Lab_Vlan01 interface.*"

The reason for this behavior is because we have not configured any default gateway, meaning that we will not go through a router to reach the target network or IP.

An important question now comes up: what will be the source IP of such an ARP request sent out by cOS Core?

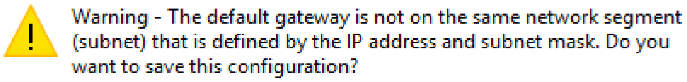
Using Local IPs with multiple networks behind the same interface

When cOS Core sends an ARP request to locate 203.0.113.16, it will use the interface address as sender. In our example it will be the interface address specified on the interface called "Lab_Vlan01", which is 192.168.1.1.

This means that the target machine receives an ARP query from a source IP that is not part of its own designated IP range (203.0.113.xx). The target machine will, in most cases, (based on how the netmask is configured) ignore and drop the request. No communication between cOS Core and the target machine will be possible as long as this problem persists.

If we attempt to configure a gateway address on our host computer using a public IP that is not part of the host's own IP range, there will be a problem.

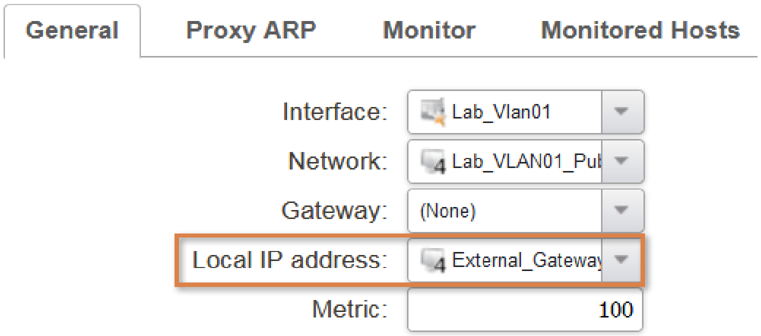
For example, Microsoft Windows will generate a warning message like the one shown in the screenshot below when attempting to do this.



Warning - The default gateway is not on the same network segment (subnet) that is defined by the IP address and subnet mask. Do you want to save this configuration?

Figure 3.16.3 Windows 10 gateway address warning

To solve this, we need to set the "Local IP" property of our newly created cOS Core route, as shown below.



Interface: Lab_Vlan01

Network: Lab_VLAN01_Pul

Gateway: (None)

Local IP address: External_Gateway

Metric: 100

Figure 3.16.4 Configuring the Local IP on a route

The "Local IP" setting performs two important tasks:

- The IP address configured for Local IP will be used as the source IP for ARP queries sent by cOS Core.
- cOS Core will respond to ARP towards the configured IP address on the target interface.

If the host computer has the IP address 203.0.113.16 assigned to it, it seems reasonable that it should use a gateway that is within the same network segment, such as the ISP router address 203.0.113.1.

Further explanation of the Local IP

Let us ask the following question: "*If cOS Core responds to the ARP request for IP 203.0.113.1, does that not cause an IP conflict with the ISP router?"*".

The answer is no, it will not. The reason why it is not a problem is because configuring "Local IP" means only that cOS Core responds to ARP messages on the designated interface. The ISP router is located behind the "EXTERNAL" interface and not Lab_Vlan01.

We are telling a "lie" about the location of the IP address so that the machine with the public IP thinks it is talking directly to the ISP router, when it fact it is talking to cOS Core.

The need for Proxy ARP

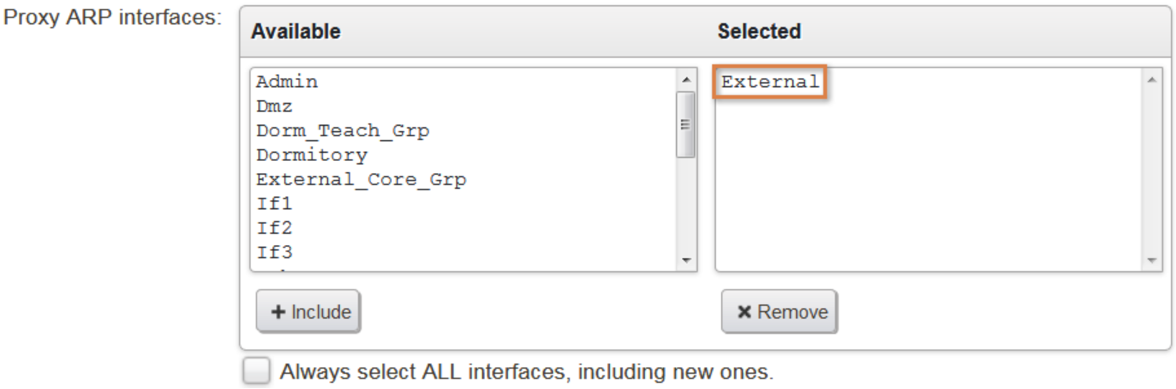
The main purpose of this recipe is to assign a public IP address to a computer behind the firewall. However, in order for that to work, we must also take into account how the ISP router behaves.

In most cases, the ISP router has a route pointing to the port or interface the firewall is located behind. This means that the ISP router will perform ARP queries to locate any hosts in the designated network segment.

If we look at the previous image of configuring the route (*Figure 3.16.4*), we have routed the public IP on the Lab_Vlan01 interface.

When the ISP sends an ARP query towards this IP (203.0.113.16) it will be received on the EXTERNAL interface and not the VLAN. Since the target host that owns the IP address is behind the VLAN and not the EXTERNAL interface, the host will not respond and communication will fail.

To solve this problem, we need to go to the Proxy ARP tab on our route and add the EXTERNAL interface as an interface on which we will Proxy ARP the network. This is shown in the next screenshot.



Proxy ARP interfaces:

Available:
```
Admin
Dmz
Dorm_Teach_Grp
Dormitory
External_Core_Grp
If1
If2
If3
```

Selected:
```
External
```

□ Always select ALL interfaces, including new ones.

Figure 3.16.5 Configuring Proxy ARP on the EXTERNAL interface

Now, cOS Core will reply to ARP queries sent by the ISP router and then forward any packet requests according to the configured routes.

Configuring IP policies

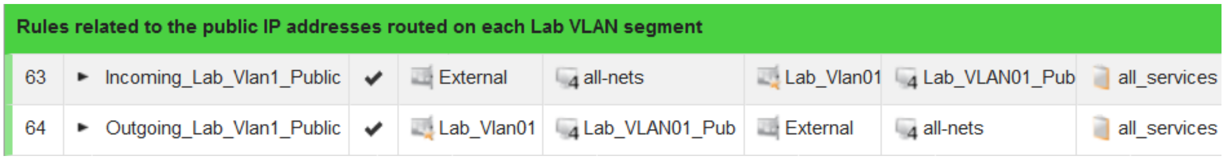
Creating IP policies for communication to/from a public IP address that is configured in this way is easier because we have no need for address translations in either direction.

Note

Since this is a lab network in the university, the policies can be extremely generous. Using a public IP internally would not normally be allowed except in special circumstances. A host exposed like this with a public IP can be a prime target for hackers.

Extra care should be taken to make sure that this host can never reach any other resources in protected networks and it should have local anti-virus scanning installed and local firewall software should be correctly configured and up to date.

An example of how the IP policies are configured is shown below.



Rules related to the public IP addresses routed on each Lab VLAN segment

63	► Incoming_Lab_Vlan1_Public	✔	External	all-nets	Lab_Vlan01	Lab_VLAN01_Pub	all_services
64	► Outgoing_Lab_Vlan1_Public	✔	Lab_Vlan01	Lab_VLAN01_Pub	External	all-nets	all_services

Figure 3.16.6 IP policies for having a public IP address on the LAB network

Optional: Traffic flow description

In order to better describe what this recipe has achieved, take a look at the next diagram.

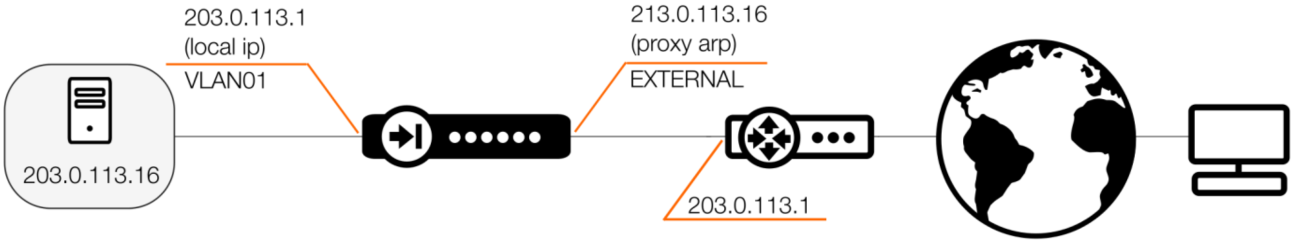


Figure 3.16.7 IP policies with a public IP address on the LAB network

The interactions that will occur in *Figure 3.16.7* above are the following:

1. A client on the Internet wants to connect to the internal server at 203.0.113.16.
2. The connection request arrives at the ISP router (IP 203.0.113.1) which performs an ARP query to locate the IP address 203.0.113.16.
3. cOS Core receives the ARP query (from IP 203.0.113.1) and replies to the ISP router saying that "I own the IP 203.0.113.16".
4. The ISP router knows where the IP address is located and starts to initiate a connection towards the target IP.
5. cOS Core now attempts to find the owner of the IP address 203.0.113.16 which it knows is located behind the Lab_Vlan01 interface. It does this by sending an ARP query using the source IP 203.0.113.1 as sender.
6. The internal server with the IP 203.0.113.16 behind the Lab_Vlan01 interface will see that the ARP query is coming from a host within its own network segment and will respond to the ARP request.

The initial part of the communication between the client and the server is now complete, connections and packets can start to flow between the two hosts.

Recipe 3.17. Bandwidth management

Objectives

The objective of this recipe is to implement a bandwidth limitation in the majority of the university networks. The purpose is to avoid a situation where a few users can allocate all available bandwidth in the University network.

We will use traffic shaping to implement a maximum bandwidth restriction based on the interface and the network. An example of what we want to accomplish is shown next in *Figure 3.17.1*.

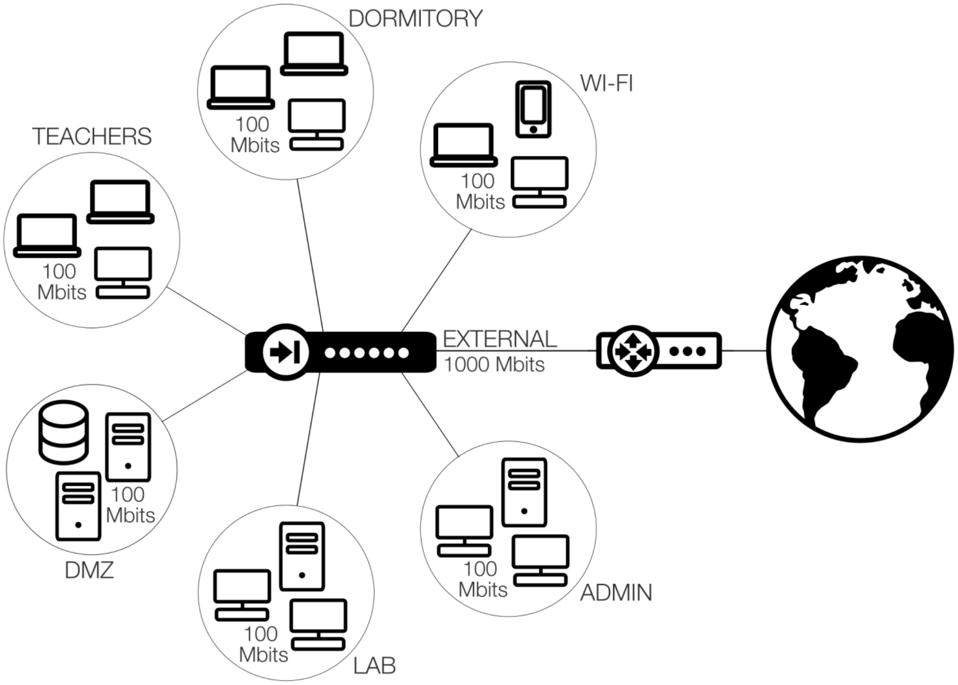


Figure 3.17.1 Setting bandwidth restrictions on interfaces and networks

Detailed Discussion

The main purpose of using traffic shaping (also referred to as traffic management, pipes or pipe rules) is to impose restrictions in our university network. For our university example, we have a fiber Internet connection of 1,000 Megabits (Mbits). By setting a limitation on each interface segment of 100 Mbits, we can avoid a situation where a single interface is able to use up all the available bandwidth.

Note

With bandwidth, "Kilo" and "Mega" are multiples of 1000, not 1024. This means a configured value of 1000 Kilobits per second on a pipe is equal to 1 Mbit.

For the interfaces LAB and DORMITORY, this is especially important because just a couple of students using peer-to-peer file sharing could easily use up a large amount of bandwidth. That could, in turn, lead to packet losses and general network disruptions.

To keep the initial scenario simple, we want to impose the restriction that each of our interfaces should, at most, be able to send and receive at a total speed of 100 Mbits per second. This means that all users behind a specific interface will be forced to share a 100 Mbit connection.

Note

In this example, we use an ISP connection with a bandwidth of 1,000 Mbits. However, in a real-life network, the 1,000 Mbits is probably never available 100% of the time. For this reason, it is recommended to always subtract an error margin of around 5 to 8 percent from the total available bandwidth when configuring traffic management.

So instead of assuming 1,000 Mbits is the total bandwidth available, it would be better to assume it is around 930 Mbits. The reason for this is that cOS Core will behave as though the configured total bandwidth is always available. If the ISP cannot actually provide this, it could lead to packet loss because cOS core will try to allocate bandwidth that actually may not be available.

Creating bandwidth pipes for each interface

To accomplish bandwidth limitation, we first need to create at least two new "pipes". A *Pipe* is a cOS Core configuration object that is used to define traffic shaping. A cOS Core object called a *Pipe Rule* then determines which traffic goes through which pipes.

Traffic shaping is configured in the WebUI under **Policies and Traffic Management**, as shown in the next screenshot.

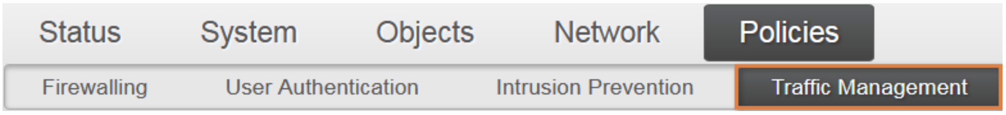


Figure 3.17.2 Traffic Management in the WebUI

Under the Traffic Shaping section, we can create the two types of traffic shaping objects, Pipe Rules and Pipes, as shown below.

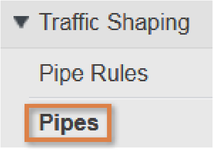
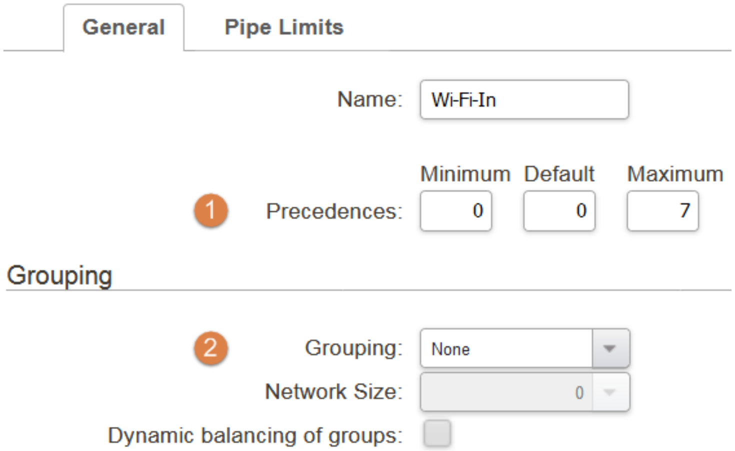


Figure 3.17.3 Traffic shaping object types

When creating the first pipe object, we are presented with the general tab, as shown below.



Name: Wi-Fi-In

	Minimum	Default	Maximum
Precedences:	0	0	7

Grouping

Grouping: None

Network Size: 0

Dynamic balancing of groups: □

Figure 3.17.4 Traffic shaping pipe's general tab

All packets that pass through cOS Core traffic shaping pipes have a **Precedence** (**1**). In this recipe, precedences will not been changed from their default values and so all packets will have the default precedence, which is zero.

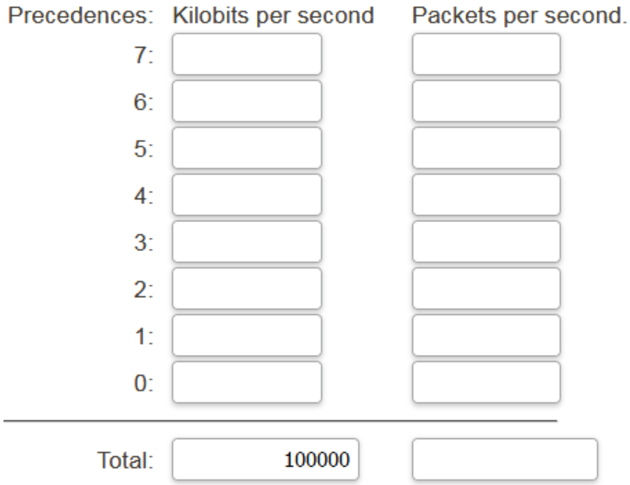
There are eight precedences which are numbered from 0 to 7. Precedence 0 is the least important (lowest priority) precedence and 7 is the most important (highest priority) precedence.

We will go into more details about precedence in a later chapter. For now, we will leave it at the default value.

cOS Core can provide a further level of control within pipes through the ability to split pipe bandwidth between users in a **Grouping** (**2**) and to apply a traffic limit and guarantee to each user. We will go into details about groupings later in this recipe. In this example, we leave it unconfigured and with the default value of *<none>*.

Pipe Limits

Now, we will move on to the Pipe limits tab, which is shown below.



Precedences:	Kilobits per second	Packets per second.
7:		
6:		
5:		
4:		
3:		
2:		
1:		
0:		
Total:	100000	

Figure 3.17.5 The pipe limits tab

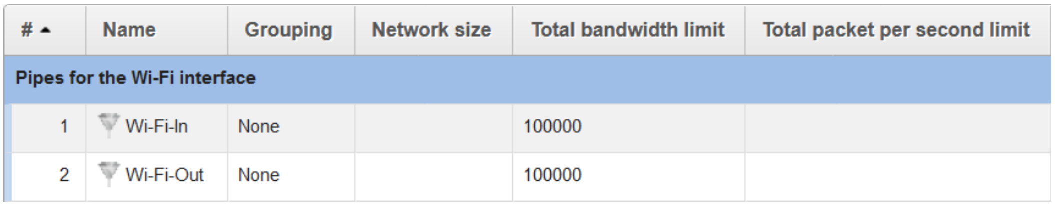
With this tab, we configure what kind of limits we want to configure in our currently selected pipe. As shown above, we have not set any values on any fields at all except the total value for the **Kilobits per second** limit.

As shown on our initial network schematic at the beginning of this recipe, the idea was to give each interface a simple bandwidth limit of 100 Mbits. Therefore we set the total pipe limit to a value of *100,000* (100 Mbits) for this pipe.

Two pipes are required, one for each direction

A common mistake regarding pipes is to forget that packets are flowing in two directions (which is often referred to as *bi-directional communication*). This means that if we only create one pipe and use that in our pipe rules for both incoming and outgoing data packets, it will be shared. This will mean, in the worst case, we could get 50 Mbits of traffic flow instead of the 100 Mbits we intended.

We therefore create two pipes, one to be used for incoming packets and another for outgoing as shown in the next screenshot.

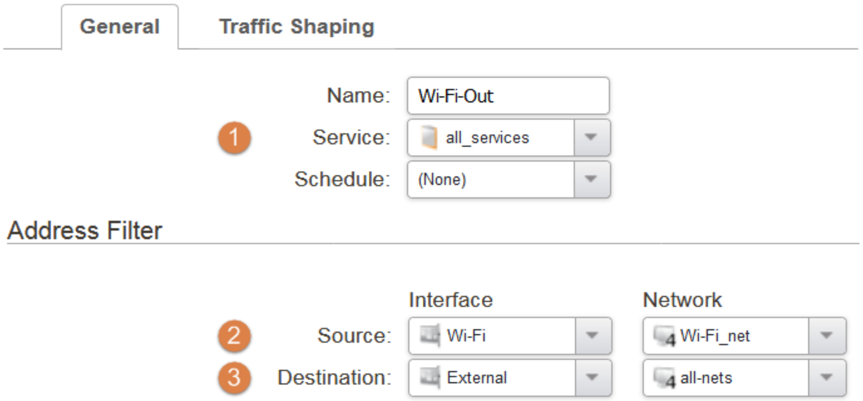


# ▲	Name	Grouping	Network size	Total bandwidth limit	Total packet per second limit
Pipes for the Wi-Fi interface					
1	Wi-Fi-In	None		100000	
2	Wi-Fi-Out	None		100000	

Figure 3.17.6 The two pipes for the Wi-Fi interface, one for each direction

Creating a pipe rule

Before we can use our newly created pipes, we must next create and configure the general properties on a Pipe Rule, as shown below.



Name: Wi-Fi-Out
Service: all_services
Schedule: (None)

Address Filter

	Interface	Network
Source:	Wi-Fi	Wi-Fi_net
Destination:	External	all-nets

Figure 3.17.7 The general properties of a Pipe Rule

With pipes, we have the option to make a pipe for specific services (**1**) such as HTTP, DNS, ICMP. That is if we want to apply bandwidth management on a very detailed level. To keep the example simple, we will be applying our traffic shaping on all available services (all ports and protocols).

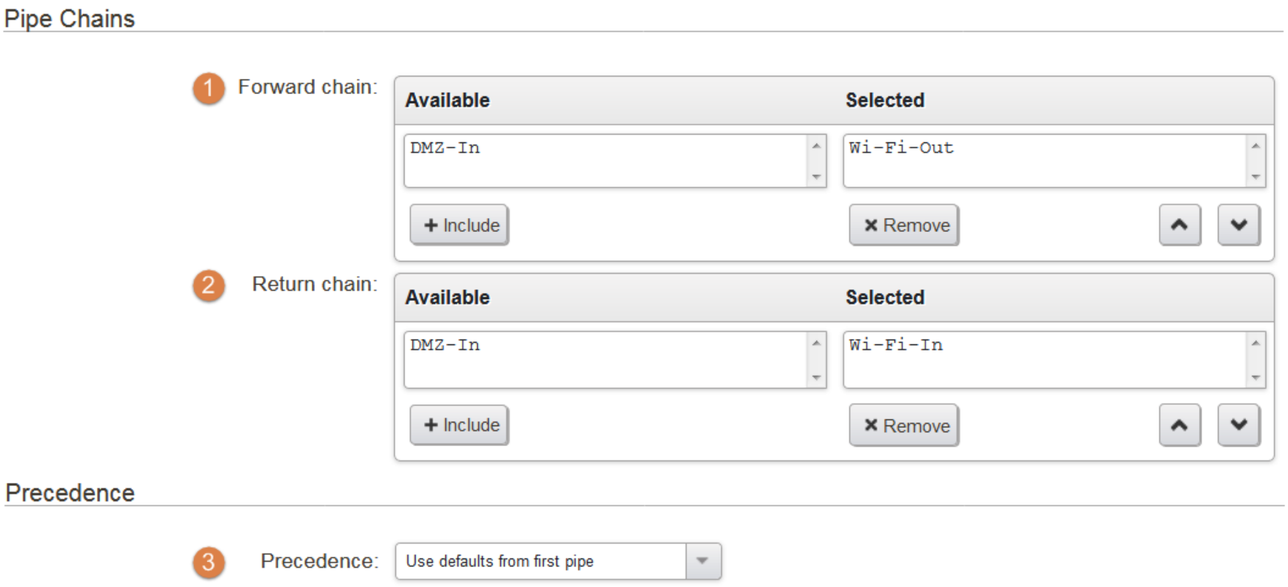
By using source (**2**) and destination (**3**) restrictions on the interface and network, we tell cOS Core that we want to apply our pipes for traffic that is initiated from the Wi-Fi interface and network towards the EXTERNAL interface (Internet).

In other words, when Wi-Fi users want to surf on the Internet we will apply a pipe restriction on the amount of total bandwidth the users behind the Wi-Fi can use when uploading or downloading data.

That is also the reason why the destination network is set to all-nets. We do not know which of the countless servers on the Internet the user will want to connect to.

Using pipes in a pipe rule

Now, it is time to use our newly created pipe in our pipe rule. For this, we go to the Traffic Shaping tab which is shown in the next WebUI screenshot.



Pipe Chains

1 Forward chain: Available: DMZ-In; Selected: Wi-Fi-Out

2 Return chain: Available: DMZ-In; Selected: Wi-Fi-In

Precedence

3 Precedence: Use defaults from first pipe

Figure 3.17.8 The pipe rule for traffic initiated behind the Wi-Fi interface

Here, traffic flow direction becomes a very important factor. When selecting which pipe to use in the forward (**1**) or return (**2**) direction, we must first be clear on what kind of traffic we are dealing with and in which direction it will flow.

Luckily, our example is very simple, so even if we by accident select the wrong pipe it would still work as both pipes are of the same size. Even so, it is important that we are clear on the traffic flow direction.

Figure 3.17.9 below, helps to better explain the traffic flow and which pipe to use in each direction.

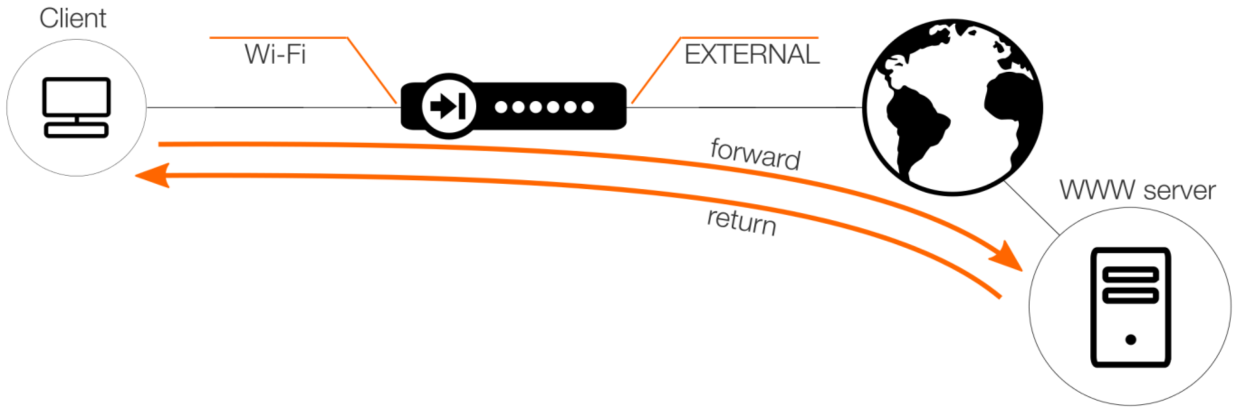


Figure 3.17.9 Using forward or return chains depending on connection initiation

The precedence setting (**3**) is used if we want to override the precedence levels defined on pipes. This setting is primarily used if several precedence levels are used on pipes in conjunction with pipe chaining. An example would be if we want to give a single IP higher precedence level than all other users in the network without creating additional pipes. We will not use this setting in this chapter.

When multiple pipe rules are needed

In our first example, we only create one pipe rule and this rule handles traffic that flows from the Wi-Fi interface towards the Internet. Normally we would create two pipe rules and the second pipe rule is used for traffic that flows in the other direction (traffic initiated from the Internet towards the Wi-Fi network).

The reason why we did not create a second pipe rule is because in our university network we have no need to create IP policies that allows traffic to be initiated from the Internet towards the Wi-Fi network. We therefore do not have a need to create a pipe rule for incoming traffic as this will never trigger. If traffic is not allowed by any IP policy, any pipes for that same traffic will never trigger.

Creating pipe rules for incoming traffic

Some of our other interfaces will definitely allow incoming connections, As an example, we will consider the DMZ interface when we create incoming pipe rules.

As we did for our Wi-Fi example above, we will create two pipes, one for incoming and one for outgoing connections, as shown next in *Figure 3.17.10*.

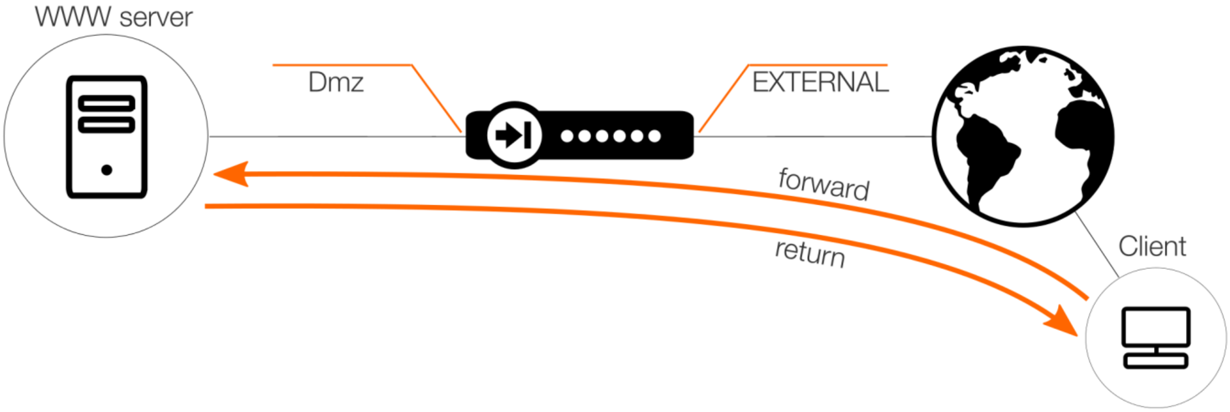
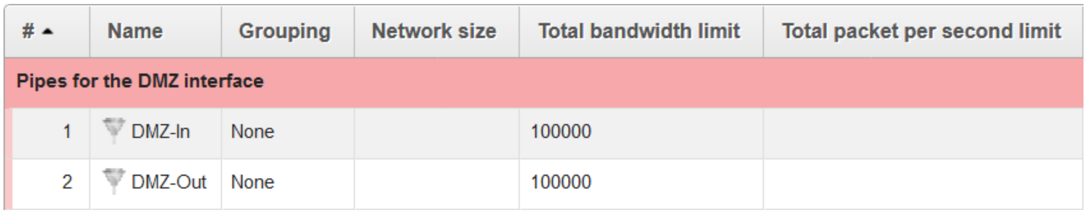


Figure 3.17.10 Forward/Return chain example for connections from the Internet

As we can see in the diagram above, the pipe direction is reversed. The forward chain is for traffic coming FROM the Internet going TOWARDS the web server. Think of the forward direction from a "who is initiating the connection" perspective. That way, it is easier to determine the forward and return chains.

When creating the pipes needed for the DMZ interface, we can simply clone the pipes we made for the Wi-Fi interface and rename them, as shown below.

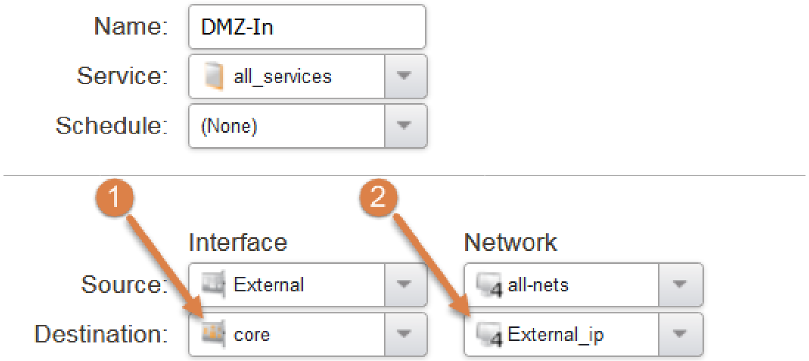


# ▲	Name	Grouping	Network size	Total bandwidth limit	Total packet per second limit
Pipes for the DMZ interface					
1	DMZ-In	None		100000	
2	DMZ-Out	None		100000	

Figure 3.17.11 Two pipe rules for the DMZ interface

Then we create two pipe rules, one for traffic that initiated from the DMZ and another rule for traffic that is initiated from the Internet. For the outgoing pipe rule, we configure them exactly as the rule for the Wi-Fi interface, except we are using the DMZ interface.

The rule that will be different is the incoming pipe rule. The first part of the incoming pipe rule is shown in the next screenshot.



Name: DMZ-In

Service: all_services

Schedule: (None)

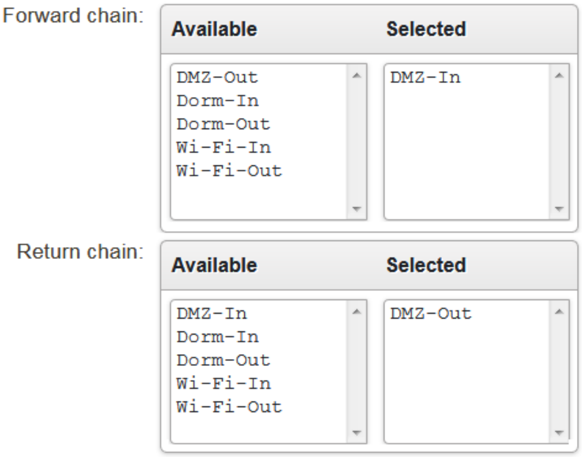
	Interface	Network
Source:	External	all-nets
Destination:	core	External_ip

Figure 3.17.12 The incoming pipe rule for the DMZ interface

The difference here is that the direction of the traffic flow is reversed. Our clients are located on the Internet and they are trying to connect to something located in the DMZ, such a web server.

For this reason, the source interface becomes EXTERNAL and the source network becomes all-nets (as we do now know which source IP the clients will have).

What can be difficult in this scenario is the fact that the destination Interface (**1** in the image above) will be Core and Destination Network (**2** in the image above) is one of the public IP addresses that cOS Core owns. The forward and return chains for incoming connections are shown below.



Forward chain:

Available	Selected
DMZ-Out Dorm-In Dorm-Out Wi-Fi-In Wi-Fi-Out	DMZ-In

Return chain:

Available	Selected
DMZ-In Dorm-In Dorm-Out Wi-Fi-In Wi-Fi-Out	DMZ-Out

Figure 3.17.13 Forward and Return chains for incoming traffic to DMZ

Destination interface of Core explained

The Destination Interface must be Core in this scenario as the client on the Internet is trying to initiate the connection FROM the Internet towards one of the IP addresses that belongs to cOS Core.

Important

Even though we use SAT in a policy to redirect the traffic to a private server on the inside, the pipe rule MUST be configured based on how traffic arrives BEFORE any address translations occur.

This is the reason why the destination interface and network for our incoming traffic is NOT the DMZ interface or the Dmz-Net object.

Note

The destination interface should be selected based on the routing decision that cOS Core makes.

Here is the difficult part, the interface is not always Core. As mentioned in *Recipe 3.15. Assigning additional interface IPs*, the destination interface can vary depending on where the IP or network is routed. Including too much or excluding some parts of the network involved in the traffic shaping configuration can cause the bandwidth allocation and calculation to be incorrect.

Modifying the bandwidth allocations based on requirements

We now have pipes and pipe rules for each interface, but are the bandwidth allocations reasonable? Is 100 Mbits enough for the Wi-Fi network? What about the DORMITORY or LAB?

Note

To keep things easy and to the point, we will view the lab network as one interface for the purposes of the example.

As we have 1,000 Mbits to play around with, we may want to revise the bandwidth allocations to be more suitable for what we expect for the various interfaces. The Wi-Fi and DORMITORY interfaces would most likely generate the most amount of traffic as the students may want to view training videos, Youtube, Netflix, and so on. That can consume a large amount of bandwidth.

We have 6 different interfaces. If we allocate 100 Mbits to each of them, we have a total of 600 Mbits, so we still have 400 Mbits to spare.

Increasing the Wi-Fi and DORMITORY bandwidth allocation would be a good idea, with 200 to Wi-Fi and 400 to the DORMITORY. This is of course a reasonable assumption but each network is unique, perhaps the DMZ interface needs more bandwidth for incoming connections so we will need to move incoming bandwidth allocations from the DORMITORY interface to the DMZ.

In the end, the decision is the administrator's.

Pipes and Stateless Policies

Pipes will not work on Stateless policies. In order to keep track of bandwidth usage between two hosts, it is a requirement that the traffic is sent through connections created by cOS Core. As Stateless Policies do not create cOS Core connections, they cannot be used in conjunction with traffic shaping.

Recipe 3.18. Dynamic bandwidth balancing

Objectives

In the previous recipe we implemented a simple bandwidth limitation per interface. This, however, only means that we tell cOS Core that users behind each interface have a bandwidth limitation, there is nothing that stops a user allocating 99% of all available bandwidth.

The objective of this recipe is to implement a bandwidth limitation method that can balance the available bandwidth dynamically between users based on their source IP to avoid one user hogging all the bandwidth for the interface behind which they are located.

Detailed Discussion

The traffic shaping covered up to this point is very basic. It is the simplest possible traffic shaping situation where we configure a fixed limit on each interface.

It has some big drawbacks. For instance, let us look at the DORMITORY network where we have implemented a 400 Mbit limit in both directions. There is no limitation per IP or user, which means that one single user could use up all the available bandwidth (in this example 400 Mbits). This will cause irritation and frustration for all the users behind the interface and lead to the situation illustrated below in *Figure 3.18.1*.

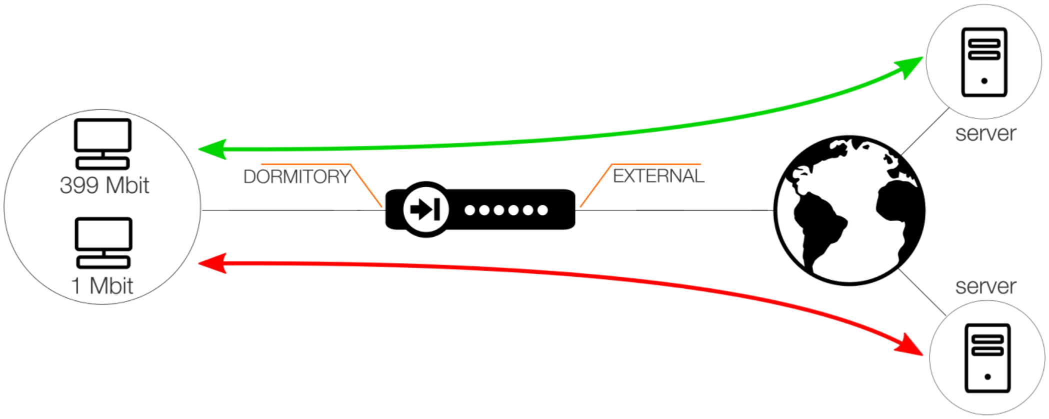
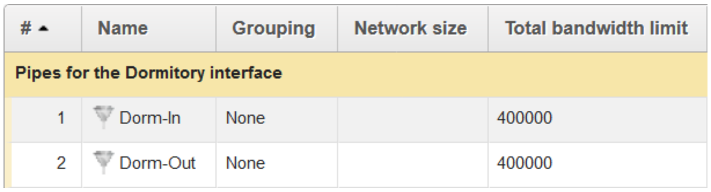


Figure 3.18.1 One user takes up nearly all bandwidth

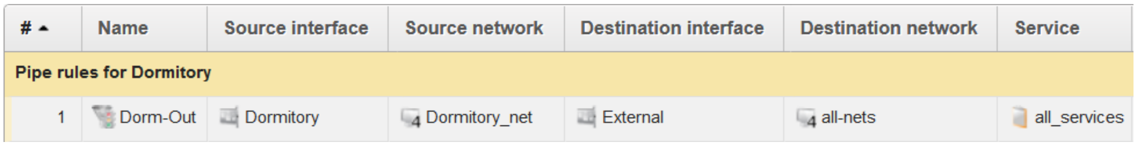
The solution to this problem is to implement grouping and dynamic balancing so that no one user can take up all available bandwidth if there are other users active in the network.

In this example, we will be using the DORMITORY interface of the university network. The reason why we choose this interface as our example is because we expect the DORMITORY network to be under the heaviest load as students will most likely want to access Youtube, Netflix and other high bandwidth services during the evenings when they unwind after a long day of lectures and study. Our current pipes and pipe rules configured for the DORMITORY interface and network are shown in the following three images.



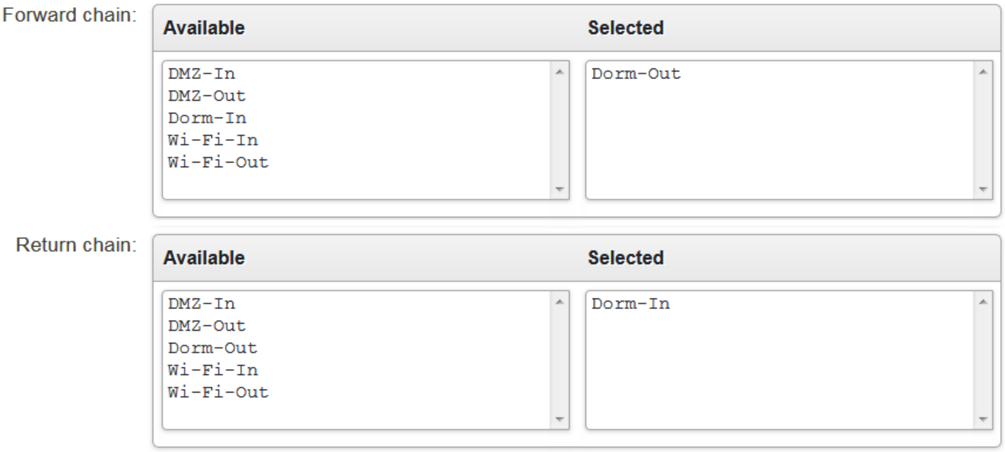
# ▲	Name	Grouping	Network size	Total bandwidth limit
Pipes for the Dormitory interface				
1	Dorm-In	None		400000
2	Dorm-Out	None		400000

Figure 3.18.2 Currently configured DORMITORY pipes



# ▲	Name	Source interface	Source network	Destination interface	Destination network	Service
Pipe rules for Dormitory						
1	Dorm-Out	Dormitory	Dormitory_net	External	all-nets	all_services

Figure 3.18.3 Currently configured DORMITORY pipe rules

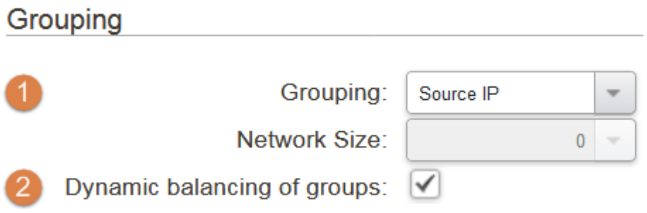


	Available	Selected
Forward chain:	DMZ-In DMZ-Out Dorm-In Wi-Fi-In Wi-Fi-Out	Dorm-Out
Return chain:	DMZ-In DMZ-Out Dorm-Out Wi-Fi-In Wi-Fi-Out	Dorm-In

Figure 3.18.4 Currently configured DORMITORY pipe rule chains

The solution: Dynamic balancing of groups

To solve the bandwidth allocation problem, we will be using an option on our pipes called Grouping (**1**) in combination of with the option Dynamic Balancing of groups (**2**) in the properties for the Dorm-Out pipe, as shown below.



Grouping

Grouping: Source IP

Network Size: 0

Dynamic balancing of groups: ☑

Figure 3.18.5 Grouping and Dynamic balancing options on the Dorm-Out pipe

By enabling these two options we are telling cOS Core two things:

- We tell cOS Core that each IP address in the DORMITORY network is to be treated as a group object.
- We tell cOS Core that each group (IP) should be balanced between each other **if** the total bandwidth available on the pipe exceeds the total limit.

Note

The grouping type will be different depending on if we are configuring the Forward or Return chain pipe. This will be discussed in more detail later on in this recipe.

If we look at the current configuration of our pipes, we see that the total limit has been set at 400 Mbits. This means that as long as we do not exceed this total bandwidth limit, no dynamic balancing will be performed.

An example of this is shown next in *Figure 3.18.6*.

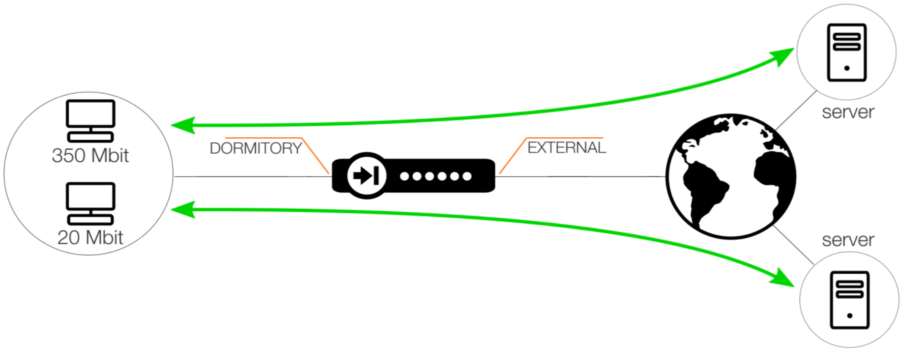


Figure 3.18.6 Each IP/client gets required bandwidth as 400 Mbits not reached

The bandwidth used by our two clients is a total of 370 Mbits. Since we have 400 Mbits to play around with, there is currently no need for cOS Core to perform bandwidth balancing.

When the total bandwidth limit is exceeded

If we have several clients that request bandwidth that exceeds the total configured limit (400 Mbits in this case), dynamic balancing will kick in and start to balance the bandwidth between the requesting clients. For example, suppose we have three hosts who request 200 Mbit each, giving a total requested bandwidth of 600 Mbit. As this is higher than the 400 Mbit limit we have, the dynamic balancing would kick in and balance the available bandwidth between these hosts so that each host would get around 133 Mbit (400 / 3 = 133). This is shown next in *Figure 3.18.7*.

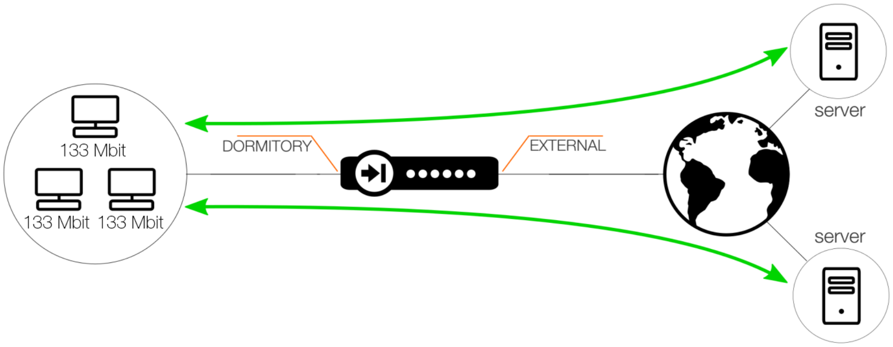


Figure 3.18.7 Dynamic balancing shares bandwidth between clients

The Forward and Return chain explained further

Here, it is important to be aware that the rules and pipes must be configured based on in which direction packets are flowing. In our DORMITORY network example, we only have only one pipe rule as we will not be allowing incoming connections being initiated from the Internet towards the DORMITORY network.

For DORMITORY, we configured one pipe rule, but a pipe rule should nearly always configured with at least two pipes. One for the forward chain and another for the return, as shown in *Figure 3.18.8*.

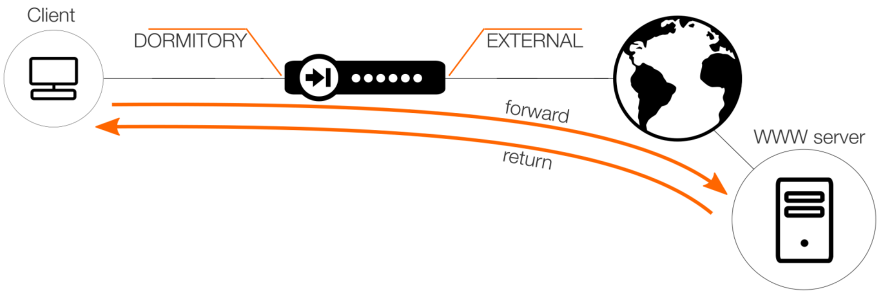


Figure 3.18.8 Forward and Return chain for the DORMITORY interface

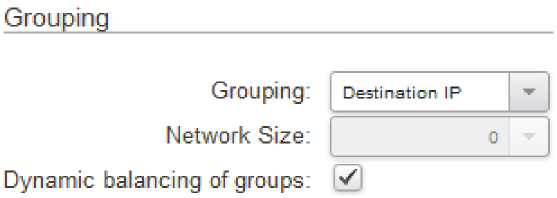
Here, cOS Core **forwards** a request/connection from a client behind the DORMITORY interface to a server on the Internet. The server then **returns** an answer to the client. This is of course true of all the traffic sent over the connection once it is established.

The grouping will be different for the incoming and outgoing pipe

The descriptions regarding grouping and dynamic balancing have so far been for the Dorm-Out pipe object. This object is used on the Forward Chain on the pipe rule as shown previously. This, however, means that we have only implemented half of the required pipe changes needed for dynamic balancing to work properly.

In order for dynamic balancing to work for the return chain, we need to modify our "Dorm-In" pipe object as well.

For the "Dorm-In" pipe, we configure grouping by destination IP, as shown in *Figure 3.18.9*.



Grouping

Grouping: Destination IP

Network Size: 0

Dynamic balancing of groups: ☑

Figure 3.18.9 Grouping and dynamic balancing for the Dorm-In pipe

The Grouping value for the Dorm-In (Return chain) will be the destination IP as we need to look at the direction the traffic is flowing from a cOS Core perspective.

For example, suppose a PC on the DORMITORY network with the IP address 10.20.10.50 connects with a web server on the Internet with the IP address 203.0.113.254. Including the port numbers, cOS Core will create a connection like this:

DORMITORY:10.20.10.50:47335 -> EXTERNAL: 203.0.113.254:80

If we then reverse this connection, it will look like this:

EXTERNAL: 203.0.113.254:80 -> DORMITORY:10.20.10.50:47335

This means that "EXTERNAL" becomes the source interface and "203.0.113.254" is the source network. "DORMITORY" and "10.20.10.50" become the destination interface and network. This is illustrated next in *Figure 3.18.10*.

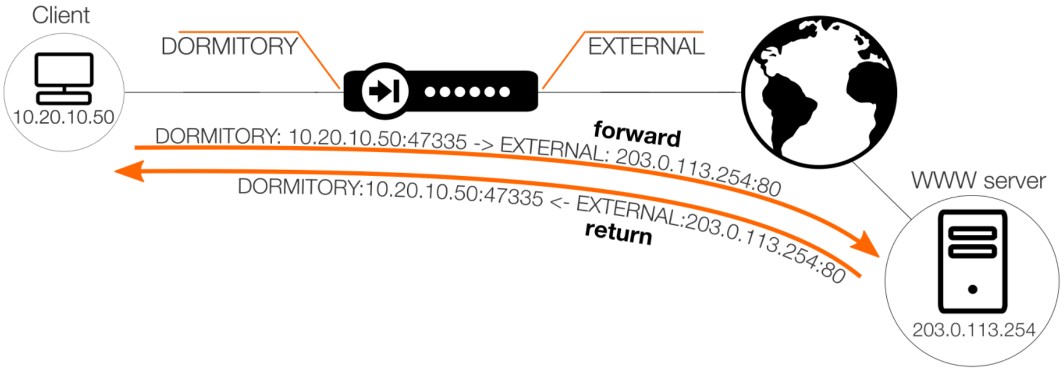

Client
10.20.10.50
DORMITORY
EXTERNAL
DORMITORY: 10.20.10.50:47335 -> EXTERNAL: 203.0.113.254:80
forward
DORMITORY:10.20.10.50:47335 <- EXTERNAL:203.0.113.254:80
return
WWW server
203.0.113.254

Figure 3.18.10 Forward and return chain example

It would not make much sense to group the return chain based on the source IP as it is a single IP. It is better to group on the destination IP as we always know it consists of the various IP addresses in our DORMITORY network and is a good separator for bandwidth distribution.

It is reasonable to assume that our clients will connect to the same target servers (such as Facebook, Youtube, Netflix). If we were to group them by the source IP for the return chain, the bandwidth distribution would still work, but just not as well.

Afterword

Thoughts from the Author

This is where we end the second edition of the cOS Core cookbook. I hope that you have developed a good practical knowledge from your reading of all the recipes that we have included so far and that your understanding of cOS Core has improved.

We have so far only scratched the surface of what we can accomplish with cOS Core. One section that I believe many find missing in this book is how to set up encrypted virtual private network (VPN) tunnels using IPsec, L2TP, PPTP, GRE and SSL.

VPN is a very extensive topic and it can be a bit overwhelming at times due to its complexity. Therefore, it is important that we first start with the basics and increase the difficulty and complexity of recipes as we progress. This cookbook focuses on basics and introduces some of the more complex scenarios but there is a lot more to explain and discuss. VPN will be included in future publications of this cookbook series.

Best regards,
Peter Nilsson.

If you have any suggestions, want to vote for a topic, send a change request or something else that you would like to see in the next cookbook, don't hesitate to contact us at *cookbook@clavister.com*.

Alphabetical Index

The Clavister

cOS Core Cookbook

2nd edition

An introduction to securing networks using a Clavister Next Generation Firewall

This is the first book about the Clavister cOS Core network operating system, where network security expert Peter Nilsson goes through the basics and also more advanced recipes on how to configure the cOS Core Firewall in various situations. This version is updated where we have moved from using IP rules to IP polices as IP policies are the future method of configuring cOS Core. Configuring cOS Core using IP Policies presents a logical and simple to use interface that combines all the necessary rule actions into one single policy, making the configuration more slim-lined requiring fewer configuration rules. The book covers a wide variety of topics related to enterprise network security operations, with the same carefully constructed real world examples now described using IP Policies.

This book is intended for anyone using the cOS Core operating system with IP Policies to secure anything from small business networks to large corporate computing infrastructure.

clavister®

CONNECT . PROTECT

ISBN 978-91-982968-6-0

9 789198 296860 >