



6WINDGate SixOS
version 6.6

6WINDGate Configuration Guide

6WIND S.A.
1, place Charles de Gaulle
78180 Montigny-le-Bretonneux
France
<http://www.6wind.com>

NOTICE

The information in this manual is provided without warranty of any kind and is subject to change without notice. **6WIND** S.A. assumes no responsibility, and shall have no liability of any kind arising from supply or use of this publication or any material contained herein.

Company and product names are trademarks or registered trademarks of their respective companies.

Copyright 2003 by **6WIND** S.A. All rights reserved.

No part of this publication may be reproduced, photocopied, or transmitted without express, written consent of **6WIND** S.A.

February 2004

P/N 10-0017-06

Table of Contents

Preface to the Configuration Guide	13
<i>Document Objectives</i>	13
<i>Conventions</i>	13
<i>References</i>	14
Chapter 1 Basics	15
<i>Basic Configuration</i>	15
<i>Using Command-Line Editing Features</i>	15
<i>Getting Help</i>	15
<i>User Accounts</i>	16
Root	16
Admin	16
Viewer	16
Logging Out	16
Changing admin Password	16
Changing viewer Password	17
Changing root Password	17
<i>Checking SixOS Software Version</i>	17
<i>Booting your 6WINDGate</i>	18
Choosing the SixOS version at boot time	18
Defining the default booting SixOS	18
<i>Updating the SixOS Software</i>	19
Chapter 2 CLI general presentation	20
<i>Definitions</i>	20
Context	20
Configuration	20
Running Configuration	20
Start Configuration	20
Non Active Configuration	20
Configuration File	20
Command File	21
Delta Mode	21
Export	21
Import	21
<i>Context Description</i>	21
<i>Prompts</i>	23
<i>CLI Syntax</i>	23
Chapter 3 Configuration, Configuration Files and Command Files	24
<i>Overview</i>	24

	<i>Configuration relationships</i>	24
	<i>Configurations files</i>	25
	Configuration File Extension	25
	Creating a New Configuration	25
	Editing a Configuration	26
	Displaying a Configuration	27
	Applying a Non Active Configuration	30
	Applying the Running Configuration in the Delta Mode	30
	Deleting a Configuration	31
	Copying a Configuration	31
	Making a Start Configuration	31
	Managing a Configuration History	31
	<i>Command Files</i>	32
	Command File Extension	32
	Editing a Command File	32
	Displaying a Command File	32
	Appending Commands to a Command File	32
	Executing a Command File	33
	<i>Exporting, Importing Configuration and Command Files</i>	33
	Exporting a Configuration File	33
	Importing a Configuration File	34
	Exporting a Command File	34
	Importing a Command File	34
Chapter 4	Configuring Remote Access	35
	<i>Introduction</i>	35
	<i>Remote Access Context</i>	35
	<i>Displaying telnet and SSH configuration</i>	35
	<i>Showing telnet and SSH Status</i>	35
	<i>Configuring Telnet Support</i>	36
	Configuring Telnet Server	36
	Telnet Client	36
	<i>Configuring SSH Support</i>	37
	Overview	37
	SSH Context	38
	Configuring the SSH Server	38
	Configuring the SSH Client	40
Chapter 5	Handling date and time	45
	<i>Configuring Date, Time and Timezone</i>	45
	<i>Configuring date and time by NTP</i>	46
	Introduction	46
	NTP Context	46
	<i>Displaying NTP Configuration</i>	46
	Configuring the NTP Client	47
	Configuration Example	48

Chapter 6	Configuring a network interface	49
	<i>Entering an interface context</i>	49
	<i>Setting the Interface Up and Down</i>	49
	<i>Listing running interfaces</i>	49
Chapter 7	Configuring broadcast interfaces	51
	<i>Assigning an IPv4 or IPv6 Address</i>	51
	<i>Address resolution (ARP and NDP handling)</i>	52
	Enabling or Disabling ARP	52
	Static ARP Entries	52
	Enabling or Disabling NDP for IPv6	53
	Modifying NDP Entries	53
	<i>Ethernet interface (eth)</i>	54
	Ethernet Context	54
	<i>Logical Ethernet Bridge group (bnet)</i>	54
	ATM Bridge	54
	VLAN Bridge	55
	Ethernet Raw Bridge	55
	IP configuration of the Ethernet bridge group	55
Chapter 8	Configuring loopback interfaces	56
	<i>Entering the loopback interfaces context</i>	56
	<i>Adding a loopback interface</i>	56
	<i>Deleting a loopback interface</i>	56
Chapter 9	Configuring point to point interfaces	58
	<i>ATM interface (atm_ptp)</i>	58
	Configuring an ATM template	58
	ATM point-to-point PVC (routed mode)	59
Chapter 10	Configuring network card controllers	61
	<i>ATM controller</i>	61
	Entering ATM controller context	61
	ATM Signaling	61
Chapter 11	Advanced configuration settings	63
	<i>Configuring Maximum Transmission Unit (MTU)</i>	63
	Presentation	63
	MTU Configuration Context	63
	Configuring MTU	63
	Configuration Example	63
	<i>Configuring TCP Maximum Segment Size(TCPMSS)</i>	64
	Presentation	64
	TCPMSS Configuration Context	64
	Configuring TCPMSS	64
	Configuration Example	64
	<i>Configuring Maximum Receive Unit (MRU)</i>	64
Chapter 12	Configuring a Serial Interface	66

	<i>Configuring Serial Physical Parameters</i>	66
	Serial Interface Context	66
	Binding a Logical Interface to a Physical Interface	66
	Configuring the Clock	66
	Configuring the Line Baud Rate	66
	Setting the Interface Up and Down	67
	Configuration Examples	67
	<i>Configuring Cisco HDLC</i>	67
	CHDLC Context	67
	Configuring CHDLC Parameters	67
	Defining CHDLC Endpoints	68
	Configuration Example	68
Chapter 13	Configuring Point To Point Protocols	69
	<i>Configuring PPP</i>	69
	PPP Context	69
	Configuring PPP parameters	69
	Defining PPP Endpoints	69
	<i>Configuring PPPoE</i>	70
	Overview	70
	PPPoE Context	70
	Setting the Interface Up and Down	70
	Configuring a PPPoE Interface	70
	<i>Configuring PPP DNS Extension</i>	71
	<i>Configuring PPP Authentication Protocols</i>	71
	Enabling CHAP or PAP	71
	Defining PAP Parameters	71
	Defining CHAP Parameters	72
	Configuration Examples	73
	<i>Rebooting the 6WINDGate 6200 series</i>	75
Chapter 14	Configuring IPv4 and IPv6 Tunneling	77
	<i>Introduction</i>	77
	<i>Definitions</i>	77
	Tunneling	77
	Automatic Tunnel	78
	IPv6 in IPv4 Configured Tunnel	78
	IPv4 in IPv6 Configured Tunnel	78
	IPv6 in IPv6 Configured Tunnel	78
	6to4	79
	ISATAP	79
	<i>Migration Techniques</i>	79
	Entering the Migration context	79
	Displaying Migration Configuration	79
	Automatic Tunnel Configuration	79
	IPv6 in IPv4 Configured Tunnel Configuration	80
	IPv4 in IPv6 Configured Tunnel Configuration	81

	ISATAP Configuration	81
	<i>IPv6 in IPv6 tunnels</i>	83
	IPv6 in IPv6 Configured Tunnel Configuration	83
	<i>DSTM</i>	84
	DSTM process	84
	DSTM applicability	85
	Configuring DSTM	86
Chapter 15	Auto-configuration functions	87
	<i>Introduction – stateful vs. stateless auto-configuration</i>	87
	<i>IPv6 Stateless Auto-configuration</i>	87
	Overview	87
	IPv6 Auto-configuration for a Router	88
	IPv6 Auto-configuration for an Auto-configurable Device	89
	IPv6 Auto-configuration for a Non Auto-configurable Device	89
Chapter 16	Configuring DHCPv6 Client and Prefix Delegation	90
	<i>Introduction</i>	90
	<i>DHCPv6 Client Context</i>	91
	<i>Displaying DHCPv6 client Configuration</i>	91
	<i>Showing DHCPv6 Client Information</i>	91
	<i>Enabling and Disabling DHCPv6 Client</i>	92
	<i>Defining Interface</i>	92
	<i>Prefix Delegation Configuration</i>	92
	Prefix Delegation request	92
	Service Level agreement ID	92
	DUID	92
	Prefix Delegation Negotiation	93
	<i>DNS Server and DNS Proxy Configuration</i>	94
	<i>Configuration Example</i>	94
Chapter 17	Configuring IP Services	96
	<i>Configuring IP forwarding</i>	96
	Enabling and Disabling IPv4 Forwarding	96
	Enabling and Disabling IPv6 Forwarding	96
Chapter 18	Configuring IP Services	5
	<i>Configuring IP forwarding</i>	5
	Enabling and Disabling IPv4 Forwarding	5
	Enabling and Disabling IPv6 Forwarding	5
Chapter 19	Configuring DNS Proxy and Client	98
	<i>Introduction</i>	98
	<i>DNS Configuration</i>	98
	Introduction	98
	Manual Configuration	98
	Dynamic Configuration	99
Chapter 20	Configuring NAT and NAT-PT	100

	<i>Introduction</i>	100
	NAT	100
	NAT-PT	101
	DNS-ALG for NAT-PT	102
	<i>NAT Context</i>	103
	<i>Displaying NAT and NAT-PT Configuration</i>	103
	<i>Showing NAT and NAT-PT information</i>	103
	<i>Configuring NAT</i>	104
	<i>NAT, Filtering and Service Flows</i>	106
	<i>Configuring NAT-PT</i>	106
	<i>NAT-PT and Filtering</i>	108
	<i>Routing the NAT-PT Prefix</i>	108
	<i>Configuration Examples</i>	108
	NAT Configuration Example	108
	NAT-PT Configuration Example	110
Chapter 21	Configuring IPv6 Mobility	113
	<i>Introduction</i>	113
	<i>Configuring IPv6 Mobility</i>	114
	MIP Context	114
	Configuring IPv6 Autoconfiguration Mechanisms	114
	Displaying Mobile IP Information	115
	Modifying the Mobile IP behaviour	116
	Managing the Mobile Node Binding Cache Entries	117
Chapter 22	Configuring SNMP	118
	<i>Configuring SNMP Support</i>	118
	Overview	118
	Enabling and Disabling SNMP	118
	SNMP Context	118
	Creating and Modifying Access Control for an SNMP Community	119
	Establishing the sysContact and sysLocation	119
	Configuring SNMP Traps	119
	SNMP Configuration Example	119
	<i>Supported MIBs</i>	120
Chapter 23	Configuring IPv4/v6 Quality of Service	121
	<i>Introduction</i>	121
	<i>Definitions</i>	121
	Class	121
	Flow	121
	PHB	121
	Expedited Forwarding	122
	Assured Forwarding	122
	Class Template	122
	<i>QoS Context</i>	122

<i>Displaying QoS Configuration</i>	122
<i>QoS Configuration Steps</i>	122
<i>Enabling QoS Management</i>	123
<i>Defining a Maximum Bandwidth</i>	124
<i>Defining a QoS Class</i>	124
<i>Defining a QoS Flow</i>	125
<i>QoS Advanced Configuration</i>	126
Class Template Presentation	126
Drop Policy Presentation	127
Configuring a Class Buffer Size	128
Configuring the Best Effort Queue Size	128
Defining the DSCP marking policy for the Best Effort	128
Configuring the interface in a single line	129
<i>Configuring the Service Flow</i>	129
Enable QoS handling for Service Flow	129
Configuring the Service Flows bandwidth	129
Configuring the Service Flow Queue Size	130
Defining the DSCP marking policy for the Service Flow	130
Configuring the Service flow class in a single line	130
<i>Monitoring QoS</i>	131
<i>QoS Configuration Examples</i>	133
Configuring Expedited Forwarding PHB	133
Configuring Assured Forwarding PHB	135
Chapter 24 Configuring IPv4/v6 IPsec	138
<i>Introduction</i>	138
<i>Definitions</i>	138
VPN	138
Static VPN	138
Dynamic VPN	138
Pre-Shared Key	139
Certificate	139
VPN Template	139
IPsec rule	139
Zone	139
Security Association	140
<i>Security Context</i>	140
<i>Displaying Security Configuration</i>	140
<i>Security Configuration Overview</i>	140
<i>Security Configuration Steps</i>	141
<i>Enabling Security</i>	142
<i>Defining Identity Parameters</i>	142
<i>Using Certificates</i>	143
Introduction	143

	Installing and Uninstalling CA Certificates and Certificates	143
	<i>Using Pre-Shared Keys</i>	145
	<i>Defining a VPN</i>	145
	<i>Defining a roadwarrior VPN</i>	147
	<i>Defining an IPsec rule</i>	147
	<i>Security Advanced Configuration</i>	149
	Defining a Static Security Association	149
	Creating a New VPN Template	151
	Deleting a User Defined VPN Template	152
	<i>IPsec and IKE Configuration Examples</i>	153
	Presentation	153
	Scenario	153
	Configuring a Static VPN	154
	Configuring a Dynamic VPN with Pre-Shared Key Authentication	156
	Configuring a Dynamic VPN with Certificate Authentication	158
	Configuring access to roadwarrior hosts with Certificate Authentication	160
Chapter 25	Configuring IPv4/v6 Packet Filtering for Firewalls	162
	<i>Overview</i>	162
	<i>Definitions</i>	163
	Rules	163
	Dynamic Rules	163
	Stateful Management	163
	<i>Filter Context</i>	163
	<i>Displaying Filter Configuration</i>	163
	<i>Showing Filtering rules</i>	164
	<i>Rule Syntax</i>	164
	Action Descriptor Syntax	164
	Packet Selector Syntax	165
	<i>Packet Filtering and Service Flow Configuration</i>	166
	<i>Packet Filtering and Migration Configuration</i>	167
	<i>Log Messages for IP Filtering</i>	167
Chapter 26	Configuring a PPP Server	169
	<i>Overview</i>	169
	<i>Configuration Contexts</i>	169
	<i>Configuring the PPP Server</i>	170
	Setting the PPP server up and down	170
	Setting the transport layer	170
	Setting the PPP server address	170
	Enabling IPv4 PPP sessions	170
	Enabling IPv6 PPP sessions	170
	Setting PPP interface MTU	170
	Setting PPP interface MRU	170
	Setting PPP interface TCPMSS	170

	Setting the pool of addresses	171
	Setting the DNS parameters	171
	Configuring PPP authentication parameters	171
	<i>Configuring A L2TP Server</i>	171
	Overview	171
	Configuration	172
Chapter 27	Configuring a AAA local database	174
	<i>The authentication local database</i>	174
	<i>Configuring a static address for a client</i>	174
	<i>Configuring IPv4 and IPv6 routes for a client</i>	174
Chapter 28	Configuring the RADIUS client	175
	<i>Access to RADIUS servers</i>	175
Chapter 29	Troubleshooting the 6WINDGate – Displaying current status	177
	<i>Overview</i>	177
	<i>Main network services</i>	177
	Displaying running interfaces	177
	Displaying running services	179
	Displaying and flushing the ARP and NDP caches	179
	Displaying Network connections	179
	Displaying Network statistics	180
	<i>Memory</i>	181
	Displaying system virtual memory	181
	<i>Show boot messages</i>	183
	<i>Routing status display</i>	184
	Displaying routing protocols status	185
	Displaying the forwarding table and protocols routing tables	189
	<i>PPP status display</i>	193
	<i>IKE status display</i>	193
	<i>IPsec status display</i>	195
	<i>IPsec statistics display</i>	197
Chapter 30	Troubleshooting the 6WINDGate – Using logging facilities	199
	<i>Configuring the logging service</i>	199
	Introduction	199
	Configuration overview	199
	Configuring logging	199
	Displaying logging information	201
	Exporting a log session	202
	<i>Routing log messages</i>	203
	Configuration overview	203
	Routing log messages configuration	203
	<i>PPP log messages</i>	204
	Configuration overview	204
	PPP log messages configuration	204

	<i>IPsec log messages</i>	205
	Configuration overview	205
	IKE log messages configuration	206
Chapter 31	Troubleshooting the 6WINDGate – Using traffic monitoring	207
	<i>Activating the traffic monitoring</i>	207
	Introduction	207
	Displaying the traffic	207
	Recording the traffic	209
	Managing the capture files	210
Appendix A	List of Acronyms	211

Preface to the Configuration Guide

This chapter discusses the objectives, organization, related documentation and conventions of the 6WINDGate SixOS Configuration Guide.

Document Objectives

This Configuration Guide describes the tasks and commands necessary to configure and manage your 6WINDGate, running the SixOS™ operating system.

Conventions

The following conventions are used to attract the attention of the reader:



Caution Means *reader be careful*. In this situation, you might do something that could result in 6WINDGate damage or loss of data.



Note Means *reader take note*. Notes contain helpful suggestions or references to materials not contained in this manual.

Command description is case sensitive. It uses the following conventions:

Convention	Description
screen	Courier plain shows an example of information displayed on the screen.
boldface screen	Courier bold is reserved for Command Line Interface key words.
<i>italics</i>	Italic text indicates arguments for which you supply values.
#	An # at the beginning of a line indicates a comment line.
< >	Angle brackets show nonprinting characters, such as passwords.
"string"	A quoted string is a quoted set of characters. It has to be used when blank characters are inserted in a string. For example, when defining the identity organization parameter as "6WIND SA", the string will include quotation marks.
string	A string is a nonquoted set of characters. For example, when setting an SNMP community string to <i>public</i> , do not use quotation marks around the string or the string will include the quotation marks.

[x]	Square brackets indicate an optional element (keyword or argument).
{x y}	Braces and vertical lines indicate a choice within a required element.
[{y z}]	Braces and vertical lines within square brackets indicate a required choice within an optional element.

References

This document does not describe routing functions configuration. This vast subject is handled in a separate document, entitled *Routing Configuration Guide*. However, troubleshooting tools for routing functions are described in the present document in chapters dedicated to status display and log messages.

Chapter 1 **Basics**

This chapter describes the basics for using the 6WINDGate Command Line Interface (CLI).

Basic Configuration

Before going further using the CLI, the installation procedures and basic configuration described in the *Installation Guide* have to be performed.

Using Command-Line Editing Features

The shell command interpreter have the ability to store typed commands in a circular memory. Typed commands can be recalled with the UP/DOWN keys and may be modified with LEFT/RIGHT/INS/DEL keys.

TAB key may be used to complete a non ambiguous partial command.

Getting Help

6WIND provides a comprehensive help on the CLI commands. This help can be invoked in different ways.

Help about a specific command is available with the following command:

```
6OS{} help command
6OS{}? command
```

Help about commands of a specific context is available with the following command:

```
6OS{}help
6OS{}?
```

To get help about the commands available in the current context, use the **help** or **?** command without parameters.

The main difference between the **help** or **?** command is that with the first one, you have an explanation about the whole comand and with the other one you have just an explanation about the keyword.

Example:

```
6OS{myconfig-sec}help vpn
Valid commands are:
```

```
vpn NAME TEMPLATENAME roadwarrior [CANAME] [one_mode_per_proto]
vpn NAME (static|TEMPLATENAME) A.B.C.D A.B.C.D [CANAME] [one_mode_per_proto]
vpn NAME (static|TEMPLATENAME) X:X::X:X X:X::X:X [CANAME] [one_mode_per_proto]
```

```
6OS{myconfig-sec} vpn ?
Valid entries at this position are:
NAME    VPN name
```

User Accounts

A **root** account and two user accounts are defined on the 6WINDGate.

Root

A **root** account is available on the 6WINDGate. This account can only be used locally through the local console port for maintenance purposes.



Caution The user must not use the **root** account, which is only available for maintenance purposes. **6WIND** will not be responsible for any trouble resulting from using this account.

Admin

In the **admin** mode, all the commands defined in the *Command Reference* are available.

Viewer

In the viewer mode, only the **display**, **logout**, **password**, **ping**, **ping6**, **show**, **slogin**, **telnet**, **telnet6** and **traceroute** commands can be used.

Editing a configuration is not possible in the viewer mode.

Logging Out

The **logout** command exits from CLI mode and login account.

Changing admin Password

The user account for configuration operations is **admin**. The default password for this account is *6windos*.

It is strongly recommended to change this default password when you first log in to the device by using the following command under the 6WINDGate prompt:

```
6OS{}password
Changing local password for admin.
Old password:<your current password>
New password:<your new password>
Retype new password:<your new password>
bsd_passwd: updating the database...
bsd_passwd: done
6OS{}
```


Changing viewer Password

The default password for **viewer** account is *6windos*. The steps to change password are the same as for **admin** account, except that you log in as **viewer** instead of **admin**.

Changing root Password

The default password for **root** account is *6windos*. The steps to change password are as follow.

```
#passwd
Changing local password for root.
New password: <your new password>
Retype new password: <your new password>
bsd_passwd: updating the database...
bsd_passwd: done
#
```



Caution It is strongly recommended to change the **root** password and keep it in a secure location.

Checking SixOS Software Version

The present document refers to the SixOS version 6.6.

The 6WINDGate is delivered with the latest SixOS version. The current version of your device can be checked using the following command:

```
6OS{ }display sys
Model: 6111
BIOS: 1.0.7 SERIAL NUMBER: 03010031 HW REV: 612100-A4
Detected cards: eth0_0 eth1_0

# OS
6WINDGate SixOS 6110 version 6.6.1 (s0)
IPsec version: S0

# Hardware
CPU: 486-class CPU
Total RAM: 67108864 (65536K bytes)

# Configuration
Last apply done with init
Uptime 0 days 00 hours 27 min 19 sec

Fri Feb 6 10:26:59 2004 (Timezone unknown)
```

If the SixOS current version is not the right one, update your device or change the SixOS to boot on.

Booting your 6WINDGate

Choosing the SixOS version at boot time

The 6WINDGate is able to store two different versions of the SixOS. This feature allows to boot on a safe SixOS if the update of a new version of the SixOS failed, for instance.

When only one SixOS is installed on the 6WINDGate, this SixOS is used for the boot.

When two SixOS are installed on the 6WINDGate, one of the SixOS is set as the default one. By default, the 6WINDGate will always boot on this SixOS. In order to boot on the other SixOS, the user will have to get into the boot menu and select the SixOS he wants to boot on.

In order to get into the boot menu, press any key except [Enter] when the following message is displayed at the beginning of the boot procedure:

```
Hit [Enter] to boot immediately, or any other key for boot menu.
```

When the boot menu is displayed, the user may choose on which SixOS the equipment will boot.

Example:

The following example shows messages that are displayed at the boot process.

```
>> FreeBSD/i386 BOOT
Default: 0:wd(0,a)/boot/loader
boot:
Console: serial port
BIOS drive A: is disk0
BIOS drive C: is disk1

FreeBSD/i386 bootstrap loader, Revision 0.7 640/59424kB
(h_debit@dodge.6wind.net, Thu Apr 25 18:16:18 CEST 2002)
\
Hit [Enter] to boot immediately, or any other key for boot menu.
```

Once any other key except [Enter] has been pressed, the following menu is displayed. Note that one of the SixOS has got the current flag, meaning that it is the one which is used by default for the boot process.

```
0 -> g-6110-6.3.1-s0.6os
1 -> g-6110-6.6.1-s0.6os (current)

Select the SIXOS [1]:0

Select the SIXOS [0]: 0
[g-6110-6.3.1-s0.6os] selected !
Booting [g-6110-6.3.1-s0.6os]...
```



Caution Some commands of the start configuration could not be correctly understood when booting on an old version of the SixOS.

Defining the default booting SixOS

When the 6WINDGate has booted, the running SixOS can be set as the default booting SixOS, by using the following command:

```
sos{ }set sixos current
```

The `set sixos current` sets the running SixOS as the default booting SixOS. The 6WINDGate will always boot with this SixOS, except if boot menu is invoked.

Updating the SixOS Software



Caution Only update files duly delivered by 6WIND can be used.



Caution It is highly recommended to save the whole equipment user data before updating the software.

To save the 6WINDGate user data, use the following command:

```
6OS{ }backup backup_URL
```

The `backup` command saves all the user data available on the 6WINDGate:

- Configuration files,
- Command files,
- SSH configuration and public keys,
- Certificate request files,
- Certificates files.

To update the SixOS software, use the following command:

```
6OS{ }update remote_updatefile_URL
```

The command installs the update file on the 6WINDGate using `tftp`, `ftp` or `scp`.



Note Before updating the SixOS, the user should download the update file from the 6WIND download server to a local file server, rather than directly updating from the 6WIND download server through the Internet. In case of an update failure, it will be easier to process update again.



Caution If two SixOS softwares are stored on the 6WINDGate, the update command will overwrite the SixOS which does not own the “current SixOS” flag. The user should check the version of the SixOS that will be overwritten before doing any update.

Example:

```
6OS{ }update tftp://server/g-6110-6.6.1-s0.6os
6OS{ }update ftp://user:pass@server/configurations/ g-6110-6.6.1-s0.6os
```

Once the `update` has successfully completed, the 6WINDGate automatically reboots.

Then, the user data previously saved by a `backup` command can be restored using the following command:

```
6OS{ }restore restore_URL
```



Caution Switching the 6WINDGate off during an update or a restore could seriously damage the 6WINDGate.

Chapter 2 CLI general presentation

This chapter gives an overview of the 6WINDGate CLI and describes the different contexts used by the CLI.

Definitions

Context

A context is an environment in which parameters can be configured. CLI commands are relevant to a context.

Configuration

A configuration describes a coherent programming of the 6WINDGate. A configuration can be edited on the 6WINDGate. A configuration includes all the statements and commands that have to be executed to configure the 6WINDGate.

Running Configuration

The **running** configuration is the one currently active on the 6WINDGate. This configuration is unique.

Start Configuration

The **start** configuration is the one that will be executed at the next 6WINDGate reboot. This configuration is unique.

Non Active Configuration

A non active configuration is any existing configuration but the running or start configuration. Several non active configurations may exist on a device at a same time.

Configuration File

A configuration file is used to transfer a configuration to or from a remote machine for editing or back up purposes (**export** and **import** commands)

Command File

A command file includes a list of commands that are executed in sequence, as if they were typed by a user. It may be used to automate configuration.

Delta Mode

The delta mode is used for dynamic parameter configuration. It makes it possible to modify the **running** configuration without stopping and restarting services whose configuration remains unchanged. As an example, adding an IP address to an interface does not stop services and does not flush interfaces configuration.

Export

Exporting a file makes it possible to transfer it from an 6WINDGate to a remote station. Exported files can be configuration files, command files, log files, certificate request files or public key files. According to the security level chosen by the user, files can be exported using the TFTP, FTP or SCP protocols.

Import

Importing a file makes it possible to transfer it from a remote station to a 6WIND equipment. Imported files can be configuration files, command files, certificate files, CA certificate files or public key files. According to the security level chosen by the user, files can be imported using the tftp, ftp or scp protocol.

Context Description

A context is an environment in which commands are entered. The contexts are organized according to a hierarchical approach represented on Figure 1. Interface contexts depend on the 6WINDGate model.

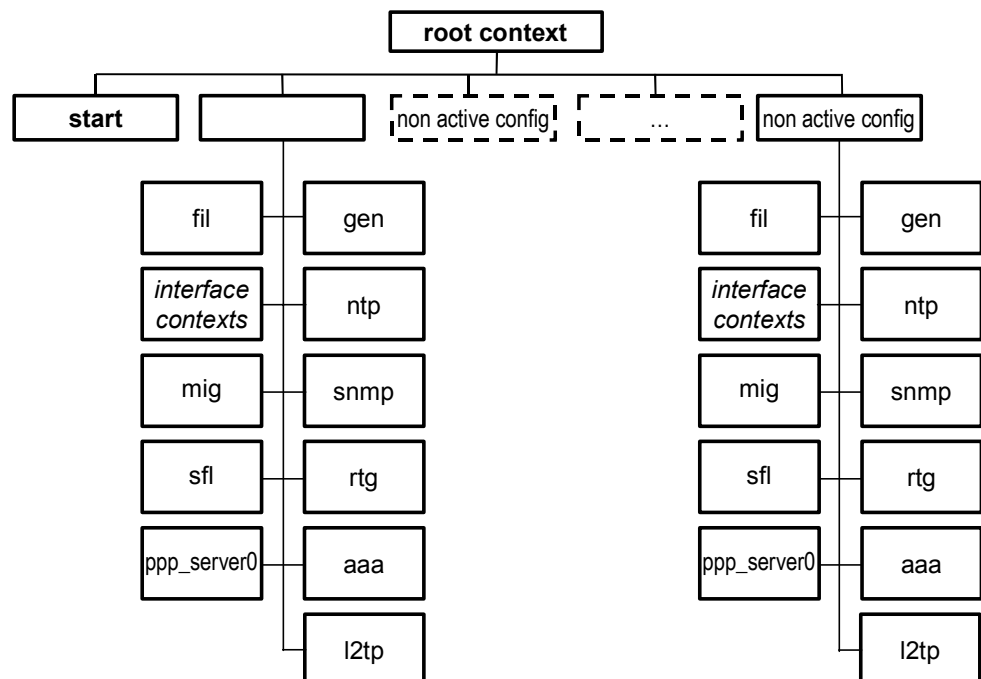


Figure 1: Context organization

The root context takes place at the top of the hierarchy. At this level the following actions can be done:

- Configuration manipulation (**edit**, **copy**...)
- Configuration, command, SSH public keys manipulation (export, import...),
- root level parameter configuration. The parameters that can be configured at this level are date and time, password, SSH parameters.

The second level is the configuration context. A configuration context is edited when entering the configuration **edit** mode. As the **start** configuration cannot be edited, configuration contexts are only defined for the **running** and all the non active configurations.

The third and last level is the function context. Several function contexts are defined in a configuration; they are listed in the following table.

Context	Description
<i>Interface contexts</i>	These contexts configure network interfaces of the 6WINDGate. Refer to related chapters for the complete list of context names.
aaa	This context enables to configure Authorization Authentication and Accounting for connection services such as a PPP server.
dhcp	This context allows to configure the router as IPv4 DHCP client.
dhcpserver	This context allows to configure the router as IPv4 DHCP server.
dhcpv6	This context allows to configure the router as IPv6 DHCP client.
dns	This context allows to configure the router as dns proxy.
fil	This context is used for IPv4/v6 packet filtering configuration.
gen	This context is used for general configuration parameters.
mig	This context is used for IPv4/IPv6 migration configuration and IP tunneling techniques.
mip	This context enables Mobile IPv6.
nat	This context is used to configure NAT and NAT-PT
ntp	This context is used to configure NTP
ppp_server0	This context enables to configure a PPP server
qos	This context is used to configure the QoS
rtg	This context is used for IPv4/v6 routing configuration.
sf1	This context is used for configuring security rules for IPv4/v6 service flows.
snmp	This context is used for SNMP configuration.

Commands are always interpreted according to the context you are in. For instance, a **display** command will not return the same result whether you are in the **rtg** or the **eth1_0** context.

Prompts

The current context is indicated in the command prompt.

For the **admin** account, the top level prompt is:

```
hostname{}
```

hostname is defined using the **hostname** command.

For the **viewer** account, the top level prompt is:

```
hostname%
```

The configuration level prompt is (**admin** account only):

```
hostname{config}
```

The context level prompt is (**admin** account only):

```
hostname{config-context}
```

All the command examples in this document are given with the prompt. The user can verify in which context the commands are entered.

Example:

```
6OS{}                # root context indicated
                    # by the hostname prompt 6OS
                    # (admin account)

6OS%                # root context indicated
                    # by the hostname prompt 6OS
                    # (viewer account)

6OS{myconfig}       # context for non active configuration
                    # myconfig

6OS{running}        # context for running configuration

6OS{myconfig-fil}   # filtering context for configuration
                    # myconfig
```

CLI Syntax

The CLI syntax is defined by a standardised grammar.

A command includes:

- An action word (**add**, **delete**, **display**, **show**) are defined to manage current objects. The action word is optional for an **add** command. Specific action words are defined for configurations, contexts and command files; they are listed in relevant sections,
- An object or service name,
- A list of parameters.

When objects like VPN templates or identity parameters require a lot of configuration parameters, the CLI provides an interactive mode. A prompt is displayed for each parameter, with its default value. The user is invited to accept the default value, by striking the return key, or to type a new value. Once the last parameter is reached, the CLI returns in the context environment.

Chapter 3 Configuration, Configuration Files and Command Files

This chapter describes how to manage user configurations and configuration files.

Overview

The 6WINDGate CLI handles files named configurations. A configuration may be considered as a file describing an expected global state of the equipment.

Editing a configuration file does not change the 6WINDGate state. A specific command (`apply` or `addrunning`) must be invoked to actually apply parameters stored in a configuration.

The 6WINDGate may be configured in two ways :

- The apply mode consists in stopping all running services, reinitializing the 6WINDGate and globally applying a configuration (via the `apply` command). If the apply succeeds, the `running` configuration will exactly reflect the applied configuration.
- The delta mode consists in directly editing the running configuration and committing changes (via the `addrunning` command). Only modifications will be applied, in order to avoid stopping services.

Configuration relationships

Figure 2 details the relationships between configurations and actions that can be performed.

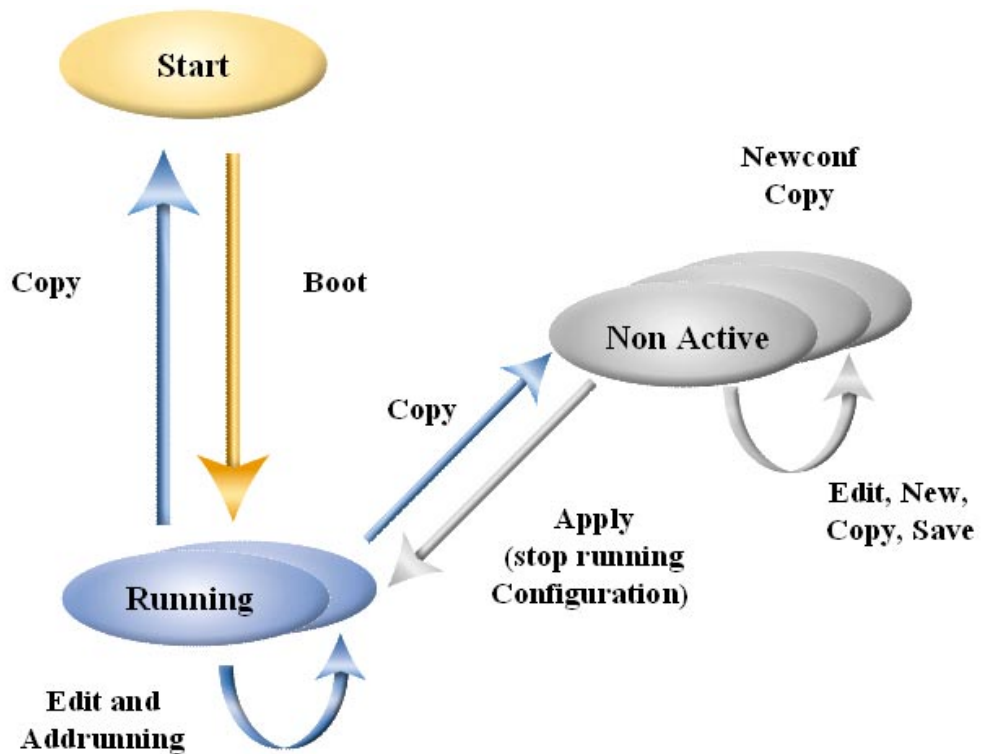


Figure 2: Configuration relationships

Configurations files

Configuration File Extension

File extension is not required for configuration files but it is recommended to use a dedicated extension (`.conf` for instance).

Creating a New Configuration

A non-active new configuration can be created starting from scratch or copied from an existing configuration. Then, the new configuration can be edited either locally on the 6WINDGate, or on a remote host.

Creating a Global Configuration From Scratch

Using the following command in the root context creates a new configuration with default values:

```
6OS{ }new conf newconf
```

If `newconf` already exists, the user is asked whether the existing configuration can be replaced or not.

Creating a Global Configuration Starting From an Existing One

Creating a new configuration is done using the following command:

```
6OS{ }copy conf config1 config2 # config1 could be running, start
```

```
# or a non active configuration
# config2 cannot be running or
# start
```

This command is detailed later in the paragraph Copying a Configuration, page 31

Editing a Configuration

Editing running or non active configuration

Whether you start from scratch or from an existing configuration, the new configuration is a non-active configuration. Then, you can get into the **edit** mode to edit the new configuration

Entering a configuration **edit** mode is done using the following command:

```
6OS{}edit conf configname
6OS{configname}
```

The **running** configuration can be edited on the 6WINDGate using the **edit** command as follow:

```
6OS{}edit conf running
6OS{running}
```

Editing a configuration can be done locally. The user can be locally or remotely connected to the 6WINDGate. In the second case, it is better to use a secure SSH connection than a telnet one.

Entering in a sub-context

Entering a function context can be done in a configuration context by typing the context name. As an example, typing the following command will enter the security context (**sec**):

```
6OS{myconfig}rtg
6OS{myconfig-rtg}
```

Entering a context name changes the current context.


```
6OS{myconfig-rtg}ntp
6OS{myconfig-ntp}
```

The user could add the required commands to configure the device. The command syntax and semantics are verified command per command.

Saving a Configuration

Using the following command in the **edit** mode saves the current modifications.

```
6OS{config1}save
6OS{config1}
```

 **Note** The **save** command cannot be used for the **running** configuration.

Exit command


The **exit** command exits the current context and returns to the upper level context. When **exit** is entered at the function level, it returns in the configuration context. When **exit** is entered at the configuration level, it returns in the root context.

If the configuration is a non-active one, the user is asked if the modifications have to be saved or not. The modifications will be saved if the user answers “y” to the save to configuration question. Answering “n” will quit the **edit** mode without saving the modifications.

If the configuration is the **running** one, all the commands entered after the previous **addrunning** are lost. In that case, the user has to confirm the **exit** command.

Example:

```
6OS{myconfig-rtg}exit           # exit the rtg context
6OS{myconfig}
6OS{myconfig}exit
WARNING: Would you like to save this config (y/n)?[y]:n
                                     # exit myconfig and discard modifications
6OS{}
```

 **Note** The *myconfig* configuration remains non active as no **apply** command has been entered yet.

```
6OS{running}exit
WARNING: Modifications will be lost. Are you sure to exit (y/n)?[n]:y
                                     # exit running and discard modifications
6OS{}
```

Displaying a Configuration

The **display conf** command, without arguments, lists the names of all the configurations available on the 6WINDGate.

Example:

```
6OS{}display conf
start
running
myconfig
6OS{}
```

Using the **display conf** command with the configuration name displays the parameters of a configuration. As shown in the following examples, the displayed configuration may be the **running**, **start** configuration or a non-active one.

```
6OS{}display conf running
6OS{}display conf start
6OS{}display conf configname
```

Example:

The new configuration is created with default values which are listed hereafter for a 6211.

```
6OS{}new conf newconf
6OS{}display conf newconf

6OS{}display conf newconf
gen
# GEN STATEMENT
hostname hurricane
domainname my.domain
enable ipv4forwarding
enable ipv6forwarding
disable telnet
enable ssh
disable http_config
```

```
# ARP TABLE
# NDP TABLE
# HOST
# NAMESERVER
# LOG
eth0_0
# IPV4 ADDRESS
# IPV6 ADDRESS
# IPV6 PREFIX
# INTERFACE STATEMENT
  intf up
  disable autoconfv6
  mtu default
  tcp4mss disable
  tcp6mss disable
  enable arp
  enable ndp
  router_advert smart 30000 1800 none 0 yes
eth1_0
# IPV4 ADDRESS
# IPV6 ADDRESS
# IPV6 PREFIX
# INTERFACE STATEMENT
  intf up
  disable autoconfv6
  mtu default
  tcp4mss disable
  tcp6mss disable
  enable arp
  enable ndp
  router_advert smart 30000 1800 none 0 yes
eth2_0
# IPV4 ADDRESS
# IPV6 ADDRESS
# IPV6 PREFIX
# INTERFACE STATEMENT
  intf up
  disable autoconfv6
  mtu default
  tcp4mss disable
  tcp6mss disable
  enable arp
  enable ndp
  router_advert smart 30000 1800 none 0 yes
loopback
# LOOPBACK INTERFACES
rtg
# IPV4 ROUTE
# IPV6 ROUTE
# DEFAULT ROUTE
  ipv4_defaultroute none
  ipv6_defaultroute none
# DYNAMIC ROUTING PROTOCOLS
  dynamic
  exit-dynamic
# LOG
mig
```

```

# 6IN4 TUNNELS
# 4IN6-6IN6 TUNNELS
# 6TO4 TUNNELS
# AUTOMATIC TUNNELS
# ISATAP_ROUTER
# ISATAP_PREFIX
# DSTM
snmp
# ROCOMMUNITY
# RWCOMMUNITY
# TRAPSINK
# TRAP2SINK
# INFORMSINK
# SNMP STATEMENT
  syslocation ""
  syscontact ""
  trapcommunity none
  authtrap enable
  disable snmp
sfl
# SRVFL STATEMENT
  ike_security clear
  ssh_security clear
  dns_security protected
  icmp_errors_security protected
  icmp_echo_out_security protected
  icmp_echo_in_security protected
  icmpv6_errors_security protected
  icmpv6_echo_out_security protected
  icmpv6_echo_in_security protected
# NMS MANAGER
# IGMP GROUPS
# MLD GROUPS
fil
# FILTER RULES IPv4
# FILTER RULES IPv6
# FILTER STATEMENT
  disable filter
# LOG
ppp_server0

# PPP SERVER STATEMENT
  service down
  ipaddress none
  enable ipv4
  enable ipv6
  disable unnumbered
  max_sessions 500
# TRANSPORT LINK
  over none
  mtu default
  tcp4mss disable
  tcp6mss disable
  mru default
# CLIENT AUTHENTICATION
  pap_ingress none
  chap_ingress none

```

```
# SERVER AUTHENTICATION
pap_egress none
chap_egress none
serverid none
# DNS
primary_dns none
secondary_dns none
# POOLS
# RADIUS MAIN SERVER
# RADIUS OPTIONAL SERVER
# LOG
debug ppp "LCP AUTH IPCP IPV6CP CCP PHYS CHAT CHAT2 IFACE L2TP"
# LOG
aaa
# AAA USERID
l2tp
# L2TP STATEMENTS
ipaddress none
disable hiding
disable sequencing
# L2TP SECRETS
exit
6OS{}
```


The commands are sorted out context per context. The context name is defined on a first line; then the commands of the context are listed, one per line. Some comments are added to help the user to find the different parameters.


Applying a Non Active Configuration

Using the following command stops the current **running** one and applies a non active configuration:

```
6OS{apply conf configname
```

Following this command, the *configname* configuration becomes the **running** one.


 **Note** The first step for applying a configuration is a syntax and semantics analysis. If this step detects some errors, it is stopped and the configuration is not applied. The **running** configuration remains unchanged.

 **Note** Applying a non active configuration is a disruptive operation. Current service will be stopped and restarted according to the new configuration definition.

Applying the Running Configuration in the Delta Mode

Using the following command applies the **running** configuration in the delta mode:

```
6OS{running}addrunning
```

 **Note** The delta mode is only possible on the **running** configuration and has to be activated in the running configuration context (**edit** mode).

The **addrunning** command is used for dynamic parameter configuration. It makes it possible to modify the **running** configuration without modifying services whose configuration remains unchanged. This is not possible with an **apply conf** command.

Note There are some restrictions to the delta mode. Modifying dynamically some features may have some consequences on other parameters.

Most of the configuration can be modified in delta mode. Nevertheless, coherence checks performed on the resulting configuration are not so extensive as in apply mode. You should take care not to create an incoherent configuration, e.g. deleting an IP address from an interface, while this address is used by another function

Deleting a Configuration

An existing configuration can be deleted using the following command:

```
6OS{}delete conf myconf
6OS{}
```

Copying a Configuration

Using the following command in the root context copies an existing configuration to a new one. It does not enter the **edit** mode:

```
6OS{}copy conf existingconf newconf
6OS{}
```

existingconf can be any configuration, including **running** or **start**.

newconf cannot be **running**. Only the **running** configuration can be copied to **start**.

If *newconf* already exists, the user is asked if the existing configuration must be replaced or not.

Making a Start Configuration

The following command will make the running configuration bootable.

```
6OS{}copy [conf] running start
```

Only the **running** configuration can be copied to the **start** configuration..



Caution Be sure that the **running** configuration has been fully tested before making it bootable.

Managing a Configuration History

Non-active configurations may be used to prepare new configurations or to store previous configurations. When applied using an **apply conf** command, a new configuration (say *newconfig*) is copied in the **running** configuration.

Command files may be used to modify the **running** configuration in the delta mode. As soon as a delta is applied, the **running** configuration differs from the initial *newconfig* one. The user can save the modified **running** configuration using the **copy** command:

```
6OS{}copy running newconfig_delta1
```

Using non-active configurations in such a way makes possible to keep a history of the running configurations.

Command Files

Command File Extension

File extension is not required for command files but it is recommended to use a dedicated extension (`.6cd` for instance).

Editing a Command File



Note A command file cannot be edited on the 6WINDGate.

To edit a command file, you must export and edit it on a remote machine. After edition, the file can be imported from the remote machine. Please refer to the paragraph Exporting, Importing Configuration and Command Files page 33

Displaying a Command File

The `display file` command, without other parameters, lists the names of all the command files available on the 6WINDGate.

Example:

```
6OS{}display file
changeroutes
addtestaddr
6OS{}
```

Using the `display file` command with a file name displays the contents of the file.

```
6OS{}display file commandfile
```

Example:

```
6OS{}display file addroutes
edit conf running
rtg
delete route 192.168.25.0/24
route 192.168.25.0/24 10.18.18.27
ipv4_defaultroute 10.18.18.1
exit
addrunning
exit
6OS{}
```

Appending Commands to a Command File


Entering the following command will make possible to append the coming commands to an existing command file:


```
6OS{}append file commandfile
```


Commands entered after an **append file** command will be appended to the *commandfile* file until entering a **close file** command.

If the *commandfile* file does not exist, it is created. If it already exists, the commands are added at the bottom of the file.

(+) is added to the prompt when the user is in an **append** mode.

 **Note** Nesting **append file** commands is not permitted.

 **Note** **append file** and **close file** commands can be only used at the root level.

Example:

```
6OS{}append file mycommand
6OS{}(+) edit conf myconfig
6OS{myconfig} (+) sec
6OS{myconfig-sec} (+) vpn myvpn cert_lite 192.0.0.1 192.0.0.2 myca
6OS{myconfig-sec} (+) exit
6OS{myconfig} (+) exit
6OS{}(+) close file
6OS{}
```

The two commands between **append file** and **close file** commands will be inserted in the *mycommand* file. If the file does not exist, it is created. If it already exists, the commands are added at the bottom of the file.

Executing a Command File

The following command executes the command file in batch mode:

```
6OS{}exec file commandfile
```

The file can be executed in all the contexts assuming that the commands are relevant to the context in which the file is executed.

Exporting, Importing Configuration and Command Files

Exporting a Configuration File


The following command exports a configuration file. The command has to be entered in the root context.


```
6OS{}export conf local_conffile remote_conffile_URL
```

TFTP, FTP or SCP protocols can be used to export configuration files.

Example:

```
6OS{}export conf myconfig tftp://server/remconfig.txt
6OS{}export conf myconfig ftp://user:pass@10.0.0.1//var/export_dir/remconfig.txt
6OS{}export conf myconfig scp://user@server/export_dir/remconfig.txt
```

 **Note** The local file name must be specified. The remote file name must be specified in the URL when using FTP or TFTP. The remote file name is optional with SCP.

 **Note** Most TFTP server implementations require that an empty file be created on the server with read and write privileges before the file can be exported.

Importing a Configuration File

The following command imports a configuration file. The command must be entered in the root context.

```
6OS{}import conf remote_conf_file_URL [local_conf_file]
```




Caution Switching the 6WINDGate off while importing a configuration file could seriously damage the equipment software.

tftp, ftp or scp protocols can be used to import configuration files.

Example:

```
6OS{}import conf tftp://server/myconfig.6cf myconfig
6OS{}import conf ftp://user:pass@10.0.0.1/configurations/myconfig.6cf
6OS{}import conf scp://user@server//users/global/conf/myconfig.6cf myconfig
```

 **Note** The local file name is optional. The remote file name must be specified in the URL.


Exporting a Command File

The following command exports a command file. The command must be entered in the root context.

```
6OS{}export file local_commandfile remote_commandfile_URL
```

Example:

```
6OS{}export file mycmdfile tftp://server/remcmdfile.txt
6OS{}export file mycmdfile ftp://user:pass@server//var/export_dir/remcmdfile.txt
6OS{}export file mycmdfile scp://user@server/export_dir/remcmdfile.txt
```

 **Note** The local file name must be specified. The remote file name must be specified in the URL.


Importing a Command File

The following command imports a command file. The command must be entered in the root context.

```
6OS{}import file remote_commandfile_URL [local_commandfile]
```

Example:

```
6OS{}import file tftp://server/mycmdfile.6cd mycmdfile
6OS{}import file ftp://user:pass@server/cmdfiles/mycmdfile.6cd
6OS{}import file scp://user@server//users/cmdfiles/mycmdfile.6cd mycmdfile
```

 **Note** The local file name is optional. The remote file name must be specified in the URL.

Chapter 4 Configuring Remote Access

This chapter describes how to configure remote access using telnet and SSH.

Introduction

The 6WINDGate can be remotely configured as it implements both telnet and SSH servers. It also implements telnet and SSH clients.

Remote Access Context

The context to configure remote access parameters is the **gen** context.

```
6OS{myconfig}gen
```

Displaying telnet and SSH configuration

To display the current configuration of **telnet** and **ssh** services, use the following commands:

```
6OS{myconfig}display gen
```

Showing telnet and SSH Status

A **show** command displays information about current status of the 6WINDGate. The displayed information are retrieved from the kernel unlike **display** commands that display only configuration information.

The **show service** command displays the current status of the services currently running on the 6WINDGate including **telnet** and **ssh**.

Example:

```
6OS{}show service
Service SSH           is active
Service SNMP         is inactive
Service NAT           is inactive
Service NATPT        is inactive
Service FILTER       is inactive
```

```
Service RIP is inactive
Service RIPng is inactive
Service OSPFv2 is inactive
Service OSPFv3 is inactive
Service BGP is inactive
Service TELNET is inactive
Service IP Forwarding is active
Service IPv6 Forwarding is active
Service IPSEC is inactive
Service DHCPSEVER is inactive
Service DHCP is inactive
Service NTP is inactive
Service Mobility node type: corresponding node

=====
ARP Table:
? (10.18.8.19) at 0:50:4:e9:c9:65 permanent [ethernet]
chanel.6wind.net (10.18.8.39) at 0:10:5a:64:ca:12 [ethernet]
? (10.18.12.23) at 0:d0:b7:bb:4f:5a permanent [ethernet]
siek.6wind.net (10.18.12.31) at 0:60:97:d6:80:40 [ethernet]
? (10.20.0.1) at (incomplete) [ethernet]
=====
NDP Table:
Address IPv6 MAC Interf
6OS{}
```

Configuring Telnet Support

Configuring Telnet Server

The 6WINDGate implements a `telnet` service. This service allows IPv4 and IPv6 hosts to connect to the 6WINDGate with a standard telnet application.

The user can enable or disable the `telnet` service by using respectively the two following commands in the `gen` context:

```
6OS{myconfig-gen}enable telnet
6OS{myconfig-gen}disable telnet
```

By default, the `telnet` service is disabled for security reasons.

Telnet Client

`telnet` and `telnet6` commands can be executed in the root context.

Information how to use `telnet` and `telnet6` can be found in the *6WINDGate SixOS – Command Reference*.

Configuring SSH Support

Overview

Secure Shell (SSH) is a protocol that provides a secure, remote connection to a remote equipment. SSH Version 1.5 is supported by the SixOS software.

The SSH server running on a 6WINDGate enables an SSH client to make a secure encrypted connection to the equipment. This connection provides functionality that is similar to an inbound telnet connection. The SSH server in the SixOS software will work with standard free or commercial SSH clients.

Additionally, the SixOS software features an SSH client that enables to communicate with a remote SSH server to export files or to log in to a remote SSH server, for instance a 6WIND equipment.

Supported applications

- Remote shell: `slogin` (SSH login client application)
- Secure file transfer: using `import` and `export` commands with the SCP protocol

Supported authentication methods

Two authentication methods are supported by 6WIND implementation:

- public key authentication with RSA key pairs
- login and password



Note If the first authentication method fails, the user is prompted for a password. Since all communications are encrypted, the password cannot be read by someone sniffing the network.

SSH connection establishment – overview

The SSH v1 protocol uses RSA public key authentication of the SSH server and client.

The SSH server is configured with:

- an SSH server key pair, which authenticates the equipment,
- a database of client public keys and/or passwords, used to authenticate remote clients connecting to local accounts.

The SSH client:

- is configured with an SSH client key pair, which authenticates the user when it connects to remote accounts,
- maintains a database of known hosts, that is to say a database associating SSH server addresses (or DNS names) to their public keys.

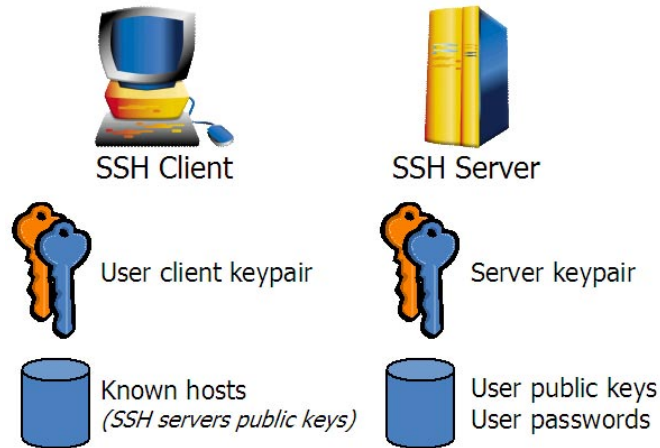


Figure 3: SSH connection establishment

An SSH connection establishment takes place as follows:

Step 1 Connection Request: the SSH client sends a connection request to the remote SSH server.

Step 2 Server authentication: the SSH server replies and authenticates itself with its SSH server key pair. The SSH client consults the local database of known hosts.

If the server identity (IP address or DNS name) is found and the associated public key is the same as the one sent by the server, the server is considered authentic and the authentication can continue,

If the server identity (IP address or DNS name) is found but the associated public key differs from the one sent by the server, either the server key pair has changed or the responding equipment is trying to usurp the SSH server identity. Hence the user gets a warning message and the connection fails,

If the server identity (IP address or DNS name) is not found in the database, the client application prompts the user and asks him if he accepts the public key sent by the responding equipment as a proof of the SSH server's identity. If the user accepts, the known hosts database is updated with this key, and the authentication can continue.

The SSH client verifies the server RSA signature. If it is valid, the connection continues.

Step 3 Client authentication: The SSH client requests a connection to a remote account on the SSH server. It authenticates itself to the SSH server, with its client key pair. The SSH server consults the database of public keys that are granted access to this account.

If the public key is found, the public key authentication can continue: the SSH server verifies the client RSA signature. If it is valid, the SSH connection is established.

If the public key is not found or the RSA authentication fails, the SSH client then tries login/password authentication. The client application prompts the user for a password. If the SSH server accepts password authentication and if the password provided by the user is the right one, the SSH connection is established. Else the SSH connection fails.

SSH Context

All the commands related to SSH parameters configuration have to be entered in the root context.

Configuring the SSH Server

Commands related to the SSH server are actually taken into account after an apply or addrunning command, though most of them are typed at the CLI root level.

Generating an SSH server key pair

Though an initial key-pair has been provided with your equipment at factory installation, it is more secure to generate a new RSA key pair for the SSH server. To perform this task, use the `ssh_server_keypair` command.

Example:

```
6OS{ }ssh_server_keypair
Generating public/private rsa1 key pair.
```

```
Your key pair already exists.
Overwrite (y/n)? y
```

```
The key fingerprint is:
1024 76:e8:83:08:57:3a:de:ee:01:77:1f:cc:5a:58:a0:7b  admin@6OS.6wind.net
```

The default key length is 1024 bits. The default SSH server key regeneration interval is 3600 seconds. To modify these parameters, refer to the *6WINDGate SixOS – Command Reference*.



Note When a new SSH server key pair is generated, SSH clients that formerly connected to the 6WINDGate will claim that the SSH server key has changed or that someone is trying to usurp its identity. The old SSH server public key should consequently be removed from their “known hosts” database.

Displaying an SSH server key pair

The `show key` command displays the server and client public keys and their fingerprints.

Example:

```
6OS{ }show key
Client public key:
1024 35 124530133164097940672699991170042656432272186026505297380193591
98090122036621255483102221114773231269842531395437856422115214410726734
03206180156078656165677050102446519832456294076574760278502870668753133
69774848241186313234477910113847867520254895773895801161758431494671806
42085788363323927652791245318057  admin@6OS.6wind.net
finger print:
1024 62:1d:be:36:3b:d2:8a:3a:9f:21:8e:c3:c0:4c:21:c2  admin@6OS.6wind.net
Server public key:
1024 35 156896570184741460625968404048211257041938863563573875828141107
97629411854118000017837501668452280847037061695622915545373959328254934
36239122389183314171810848522096277175573026166854733569580048836615049
65560819879815744533310591590205471489198134453666609751521390046350543
607602932135096292690269298596023  admin@6OS.6wind.net
finger print:
1024 76:e8:83:08:57:3a:de:ee:01:77:1f:cc:5a:58:a0:7b  admin@6OS.6wind.net
```

Choose clients authentication method

SSH clients may authenticate themselves by public key authentication (RSA key pairs) or by the classical login/password method. To increase the security level, you can disable password authentication. The following command configures this feature:

```
6OS{ }passwd_auth {yes|no}
```

Public key authentication is always possible, regardless of the value of the `passwd_auth` command.

Add authorized clients public keys

To enable remote clients to authenticate themselves by public key authentication, their RSA public keys must be authorized.

Two methods enable to add an authorized client public key: importing or manually entering it.

Importing a client public key

To be imported, the client public key must be stored on a file server, and can be imported by tftp, ftp or scp protocols.

```
6OS{}import key remote_keyfile_URL [local_keyfile]
```

Example:

```
6OS{}import key tftp://server/client1_pub.key
6OS{}import key ftp://user:pass@server/client1_pub.key
6OS{}import key scp://user@server/client1_pub.key
```



Note For security reasons, it is recommended to transfer the key from a trusted local network.

Manually entering a client public key

The `enterpublickey` enables to manually enter an SSH public key, for instance by copy-pasting it from a console.

```
6OS{}enterpublickey local_keyfile
```

Example:

```
6OS{}enterpublickey vade
Enter key:
1024 35
139045048395844991731801565353728770169874620931635899851158818990943036911519952
641530347230424995731789246028355764893490094140319363882336211224702808041317879
076618360062033632602159651681778277783563809749077442621544207675405160405361221
843117438488835950887872772707890382625629817793940565600328298771
admin@vade.6wind.net
Public key vade created
```

Storing a public key in the authorized keys database

The keys must then be stored in the authorized public keys database.

```
6OS{}publickey local_keyfile
```

The `delete publickey` and `delete key` respectively unregister a public key from the authorized public keys database and remove the public key file from the 6WINDGate.

Example:

```
6OS{}publickey client1_pub.key
```

Enabling or disabling the SSH server

The ssh server is enabled by default. To start or stop it, use the following commands:

```
6OS{myconfig-gen}{enable|disable} ssh
```

Configuring the SSH Client

The SSH client enables a user to import or export files via the SCP protocol, and to log on a remote SSH server, for instance a remote 6WINDGate.

Generating an SSH client key pair

To use public key authentication, a RSA key-pair must be created on the 6WINDGate for the `admin` user.

In order to create an SSH client RSA key-pair, log in as `admin` and use the `ssh_client_keypair` command.

Example:

```
6OS{}ssh_client_keypair
Generating public/private rsa1 key pair.

Your key pair already exists.
Overwrite (y/n)? y

The key fingerprint is:
1024 62:1d:be:36:3b:d2:8a:3a:9f:21:8e:c3:c0:4c:21:c2 admin@OS.6wind.net
```

Once the client key pair is generated, the public key can be displayed or exported, in order to be installed on a remote SSH server.

Displaying an SSH client key pair

The `show key` command displays the server and client public keys and their fingerprints.

Example:

```
6OS{}show key
Client public key:
1024 35 124530133164097940672699991170042656432272186026505297380193591
98090122036621255483102221114773231269842531395437856422115214410726734
03206180156078656165677050102446519832456294076574760278502870668753133
69774848241186313234477910113847867520254895773895801161758431494671806
420857883633323927652791245318057 admin@6OS.6wind.net
finger print:
1024 62:1d:be:36:3b:d2:8a:3a:9f:21:8e:c3:c0:4c:21:c2 admin@6OS.6wind.net
Server public key:
1024 35 156896570184741460625968404048211257041938863563573875828141107
97629411854118000017837501668452280847037061695622915545373959328254934
36239122389183314171810848522096277175573026166854733569580048836615049
65560819879815744533310591590205471489198134453666609751521390046350543
607602932135096292690269298596023 admin@6OS.6wind.net
finger print:
1024 76:e8:83:08:57:3a:de:ee:01:77:1f:cc:5a:58:a0:7b admin@6OS.6wind.net
```

Exporting an SSH client key pair

The SSH client public key can be exported to a remote server via ftp, tftp or scp protocols:

Example:

```
6OS{}export key user tftp://server/identity.pub
```

Logging in to an SSH server

To log on a remote SSH server, use the `slogin` command:

```
6OS{}slogin [user@]host
```

Where:

- `user` is an optional remote account name, `admin` by default.
- `host` is the DNS name or IP address of a remote SSH server

On the first connection to a remote SSH server, the user will be prompted if he accepts the remote server public key as a proof of its identity:

Example:

```
6OS{}slogin 10.16.0.55
The authenticity of host '10.16.0.55 (10.16.0.55)' can't be established.
RSA1 key fingerprint is d1:af:3b:02:4a:1f:de:62:2b:3c:d7:2b:67:58:64:bd.
Are you sure you want to continue connecting (yes/no)? yes
```

Warning: Permanently added '10.16.0.55' (RSA1) to the list of known hosts.

Use the fingerprint to verify that the key provided by the server is valid (this fingerprint can be obtained offline from the administrator of the SSH server). On subsequent connections, the `slogin` command will no longer ask this question. It will permanently trust this public key as a proof of this server's identity.

Displaying known SSH servers public keys

Known SSH servers public keys may be displayed via the `display remote_keyhost` command:

```
6OS{}display remote_keyhost [host]
```

Where:

- `host` is an optional SSH server DNS name or IP address. If provided, the public key of this server will be displayed. If not provided, only the list of known servers will be displayed.

Example:

```
6OS{}display remote_keyhost
10.23.2.202
3ffe:304:124:2302::202
10.16.0.51

6OS{}display remote_keyhost 10.23.2.202
public key:
1024 35 147933687600564303446617037335480645721670252884730607393837733
60033109131352375650546392044854287509750518349411691125700284948229927
49232193725557231038930332704741663877636851358756258363815393504496968
89943659932788315019930301404118484391842963301949093744087401670347543
041150489131588197886972302003133 10.23.2.202
finger print:
1024 6f:28:8a:7a:52:ea:b1:98:7a:1d:26:f7:93:5b:d6:1d 10.23.2.202
```

Removing known SSH servers public keys

To stop trusting a remote SSH server public key, use the `delete remote_keyhost` command.

```
6OS{}delete remote_keyhost host
```

Where:

- `host` is the DNS name or IP address of the remote SSH server

This command is particularly useful when an SSH server changed its RSA key pair. The user who tries to connect will receive the following message:

Example:

```
6OS{}slogin 10.16.0.55
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@      WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!      @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
```

Someone could be eavesdropping on you right now (man-in-the-middle attack)!

It is also possible that the RSA1 host key has just been changed.

The fingerprint for the RSA1 key sent by the remote host is

```
63:e8:bc:82:ea:27:ce:12:2e:d9:f8:7f:49:6f:8f:5c.
```

You must delete the old key (from 10.16.0.55) to accept the new one.

Host key verification failed.

Importing or exporting via the SCP protocol

The 6WINDGate enables to import or export files via the SCP protocol. The user just specifies the `scp` protocol in URLs. Necessary configuration of the SCP client is exactly the same as for `slogin`.

Contrary to `ftp` URLs, the remote user password cannot be specified in `scp` URLs.

Example:

```
6OS{}export key user scp://user@server/identity.pub
6OS{}import file scp://user@server.acme.com/conf/newconf.6cf
```



Note

The 6WINDGate SSH client can chose to install its public key on an SSH server by exporting it. For that purpose, it can use the SCP protocol, as long as it avoids a chicken-and-egg situation: the user cannot authenticate itself by RSA authentication as long as his public key is not installed on the remote SSH server. The solution may be to first use password authentication or to use another protocol to export the key to the server.

Chapter 5 Handling date and time

Configuring Date, Time and Timezone

The command **date** used without parameters allows displaying the system date, time and time zone. For example:

```
6OS{ }date
Tue Dec 12 19:04:31 2001 Europe/Paris
```

The command **date** is used for setting the date and time with the following representation:

```
6OS{ }date [[[[[cc]yy]mm]dd]HH]MM[.ss]
cc          Century (either 19 or 20).
yy          Year in abbreviated form (e.g. 89 for 1989, 06 for 2006).
mm          Numeric month, a number from 1 to 12.
dd          Day, a number from 1 to 31.
HH          Hours, a number from 0 to 23.
MM          Minutes, a number from 0 to 59.
ss          Seconds, a number from 0 to 61 (59 plus a maximum of two leap seconds).
```

Everything but the minutes is optional.

Example:

```
6OS{ }date 8506131627          # sets the date to June 13, 1985,
                               # 4:27 PM

6OS{ }date 1432                # sets the time to 2:32 PM, without
                               # modifying the date.
```

The **timezone** command initializes the time conversion information. The command has the following format:

```
6OS{ }timezone REGION CITY
```

Where:

- *region* is your local region
- *city* is your local city

If no **timezone** parameters are defined, the Coordinated Universal Time (CUT) is used by the equipment.

Example 1:

```
6OS{ }timezone Europe Paris
6OS{ }
```

Example 2:

```
6OS{}timezone Europe New_York
ERROR(108007) : Wrong name: (Europe/New_York). We assume Europe-Paris.
If it is not convenient, start again.
Local date is:
Tue Jun 12 10:03:32 2001 Europe/Paris
6OS{}
```



Note In order to list all available region or city names, press twice on the <TAB> key.

Configuring date and time by NTP

Introduction

The internet device on the network contains an internal system clock that is used to maintain accurate time for the equipment. The manual date configuration is different between each equipment on the network. Moreover, the date is rarely reset at regular intervals. Then, the clock is subject to exterior elements that can drift as much as several seconds each day. NTP solves this problem by automatically adjusting the time of the internet devices so they are synchronized within milliseconds.

The 6WINDGate implements Network Time Protocol client (NTP), documented in RFC 1305. This protocol allows to time -synchronize a network of machines with remote NTP servers.

NTP runs over UDP (port 123) which in turn run over both IPv4 and IPv6. The 6WINDGate NTP client complies with versions 1 to 4 of the protocol.

The communications between the NTP client and server, called “associations” are usually statically configured. It is thus necessary to define the IP addresses of the machines with which the equipment will establish an association. However, to simplify the configuration, it is possible to configure the 6WINDGate to simply receive IP broadcast messages. In this last case, the accuracy of timekeeping will be weaker because the information flow is one-way only.

Moreover, the 6WINDGate can secure the time configuration with an authentication of messages coming from NTP servers, to avoid the accidental or malicious setting of incorrect time

NTP Context

NTP parameters can be configured in the `ntp` context.

```
6OS{myconfig}ntp
```

Displaying NTP Configuration

The NTP configuration can be displayed using the following commands:

```
6OS{myconfig}display ntp
```

or

```
6OS{myconfig-ntp}display
```

Configuring the NTP Client

Enabling and Disabling NTP

NTP can be enabled and disabled using the following commands:

```
6OS{myconfig-ntp}enable ntp
6OS{myconfig-ntp}disable ntp
```

Configuring NTP Default Version

The NTP default version value is used by all NTP commands except if they override it.

```
6OS{myconfig-ntp}defaultversion number
```

where:

- *number* is the NTP version number. The valid values are 1 to 4. The default value is 4.

Specifying NTP boot remote server

Specifying the NTP server to be used at boot time is mandatory and can be done using the following command:

```
6OS{myconfig-ntp}bootserver {none | {{dhcp | address} [key number] [version version]}}
```

where:

- **none** is the initial value; NTP service requests a bootserver address,
- **dhcp** specified that the bootserver address will be got by the dhcp client,
- *address* is the IP address of the remote NTP server,
- **key number** is the md5 key number ranging from 1 to 65536 (NTPv3 or NTPv4), The number key is required if you want to authenticate your session with the NTP remote server,
- **version version** is the NTP version number. The accepted values are 1 to 4.


Configuring NTP association(s)


In order to keep time accuracy, it is necessary to specify which NTP server(s) have to be used after boot time. The following commands can be used respectively for client/server, broadcast and multicast modes:

```
6OS{myconfig-ntp}remoteserver address [key number] [version version] [prefer]
6OS{myconfig-ntp}broadcastclient [key number] [version version]
6OS{myconfig-ntp}multicastclient address [key number] [version version]
```

where:

- *address* is the IP address of the remote NTP server,
- **key number** is the md5 key number ranging from 1 to 65536 (NTPv3 or NTPv4). The number key is required if you want to authenticate your session with the NTP remote server.
- **version version** is the NTP version number. The accepted values are 1 to 4
- **prefer** marks the peer as the preferred peer that provides synchronization

 **Note** In broadcast mode, NTP broadcast messages are received by any local interface

 **Note** In multicast mode, *address* cannot be a host name

Deleting NTP association(s)

To delete the NTP associations, the following commands can be used:

```
6OS{myconfig-ntp}delete remoteserver address
6OS{myconfig-ntp}delete broadcastclient
6OS{myconfig-ntp}delete multicastclient address
```

Configuring NTP authentication

To configure NTP authentication, the following command can be used:

```
6OS{myconfig-ntp}{enable|disable} authentication
```

Default value for `authentication` is `enable`.

Defining an authentication key

The following command adds an authentication key:

```
6OS{myconfig-ntp}key number value
```

The following command deletes an authentication key:

```
6OS{myconfig-ntp}delete key number [value]
```

where:

- `number` is the md5 key number ranging from 1 to 65536 (NTPv3 or NTPv4),
- `value` is a 1-32 ASCII string.

Configuration Example

```
6OS{myconfig-ntp}enable ntp
6OS{myconfig-ntp}bootserver 1.2.3.4
6OS{myconfig-ntp}key 1 "azerty"
6OS{myconfig-ntp}key 2 "qwerty"
6OS{myconfig-ntp}remoteserver ntp.mynetwork.com key 1 version 3
6OS{myconfig-ntp}remoteserver 3ffe::1 key 2 version 4
6OS{myconfig-ntp}delete key
```


Chapter 6 *Configuring a network interface*

Interfaces are objects that represent network access points. They can be divided in two main categories:

- physical interfaces (Ethernet network port, serial network port) and logical interfaces (ATM PVC, PPPoE interface, tunnel, loopback interface, ethernet bridge...),
- logical interfaces (a.k.a. pseudo-interfaces) alleviate the administrator's job by enabling him to handle network connections the same way as physical network ports. This facility is essential for routing for instance.

Network interfaces have generic properties (such as IP addresses) and functions (such as turning them on or off), and some other properties or functions specific to their type (e.g. authentication information or a lower protocol specification). This chapter defines generic properties and functions.

Entering an interface context

The administrator enters an interface context by typing its name from a configuration context.

Example:

```
6OS{myconfig}eth1_0
6OS{myconfig-eth1_0}
```

Setting the Interface Up and Down

An interface can be set up and down using the following command:

```
6OS{myconfig-eth0_0}intf {up|down}
```

By default the interface is configured as *up*.

Setting the interface up, means that the system will try to make it run (which can fail for instance because an ethernet cable is not plugged or a PPPoE negotiation has not succeeded yet).

Setting the interface down means that the system will not activate the interface, and close the network connection if necessary (e.g. for a PPPoE interface).

Listing running interfaces

The `show interface` command lists running interfaces and some of their characteristics.

Example:

```
6OS{}show interface
eth0_0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:90:fb:04:07:86
    media: Ethernet autoselect (10baseT/UTP <full-duplex>)
eth1_0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.16.0.187 netmask 0xffffffff broadcast 10.16.0.255
    ether 00:a0:24:70:ae:5d
eth2_0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.23.4.104 netmask 0xffffffff broadcast 10.23.4.255
    ether 00:40:33:93:ef:70
```

Automatic tunnel:

```
tu0: flags=1041<UP,RUNNING,LINK0> mtu 1480
    inet6 ::127.0.0.1 /96
    lladdr 7f:00:00:01
```

Chapter 7 *Configuring broadcast interfaces*

Broadcast interfaces are interfaces directly connected to a broadcast network. They are configured with one or more IP addresses and their associated prefix lengths. Broadcast interfaces need an adaptation layer between the IP protocol and the link layer protocol, in order to map IP addresses to link layer addresses. IPv4 uses ARP (Address Resolution Protocol) and IPv6 uses NDP (Neighbor Discovery Protocol) to dynamically perform this mapping.

Assigning an IPv4 or IPv6 Address

The following commands can be used to assign an IPv4 or IPv6 address to an interface:

```
6OS{myconfig-interface}ipaddress ipv4address/prefixlen
6OS{myconfig-interface}ipaddress ipv6address/prefixlen
```

The command is the same for IPv4 and IPv6 addresses. Several addresses may be configured on the same interface (automatic aliasing is performed as soon as several addresses are declared on the same interface).

An assigned IPv4 or IPv6 address can be deleted using the following commands:

```
6OS{myconfig-interface}delete ipaddress ipv4address/prefixlen
6OS{myconfig-interface}delete ipaddress ipv6address/prefixlen
```

Example:

To configure and activate an IPv4 address, use the following command:

```
6OS{myconfig-eth1_0}ipaddress 10.12.37.1/24
```

To configure and activate the following IPv6 address with a 64 prefix length, use the following command:

```
6OS{myconfig-eth1_0}ipaddress 3ffe:305:107:101::3/64
```


Aliases make it possible to add IPv4 or IPv6 alias entries when you want a single interface to have multiple IP addresses.

```
6OS{myconfig-eth1_0}ipaddress 10.0.0.55/24
6OS{myconfig-eth1_0}ipaddress 10.16.0.56/16

6OS{myconfig-eth1_0}ipaddress 3ffe:304:107:101::2/64
6OS{myconfig-eth1_0}ipaddress 3ffe:304:107:120::2/64
```

To delete an IPv4 or IPv6 address, use the following command:

```
6OS{myconfig-eth1_0}delete ipaddress 10.12.37.1/24
6OS{myconfig-eth1_0}delete ipaddress 3ffe:304:107:101::3/64
```

 **Note** Two addresses on the same 6WINDGate cannot share the same prefix and prefix length, even on the same interface (for example, `ipaddress 10.0.0.55/24` and `ipaddress 10.0.0.32/24` cannot be simultaneously configured on the same 6WINDGate. However, `ipaddress 10.0.0.55/24` and `ipaddress 10.0.0.32/32` can).

Address resolution (ARP and NDP handling)

Enabling or Disabling ARP

The 6WINDGate SixOS makes it possible to enable or disable ARP (Address Resolution Protocol) on each Ethernet interface available in the device.

If ARP is disabled, it is possible to add ARP entries in the ARP table to control directly the protocol. This could be useful for security purposes particularly on the public interface of the 6WINDGate.

To enable ARP on an Ethernet interface, the following command has to be entered:

```
6OS{myconfig-ifname}enable arp
```

To disable ARP on an Ethernet interface, the following command has to be entered:

```
6OS{myconfig-ifname}disable arp
```

By default, ARP is enabled on all the Ethernet interfaces.

By default, NDP is enabled on all the Ethernet interfaces.


Static ARP Entries


The following command can be used to modify the Internet-to-Ethernet address translation table entries used by ARP, use:


```
6OS{myconfig-gen}arprentry host etheraddress
```

where:

- *host*: has to be specified by an IPv4 address, using Internet dot notation.
- *etheraddress*: Ethernet address is given as six hexa bytes separated by colons.

 **Note** The ARP table is global for the different Ethernet interfaces of the device.

 **Note** The user should avoid modifying the ARP table without disabling the ARP protocol

 **Note** The IPv4 address must be a neighbour address (i.e: share a common prefix with one of the interfaces).

To delete ARP entries, use the following command:

```
6OS{myconfig-gen}delete arpentry host etheraddress
```

All the table entries can be deleted using the following command:

```
6OS{myconfig-gen}delete arpentry all
```

Example:

```
6OS{myconfig-gen}arpentry 20.0.0.144 00:10:5A:64:CB:2B
6OS{myconfig-gen}delete arpentry 20.0.0.144 00:10:5A:64:CB:2B
6OS{myconfig-gen}delete arpentry all
```

Enabling or Disabling NDP for IPv6

The 6WINDGate SixOS makes it possible to enable or disable NDP (Neighbor Discovery Protocol) on each Ethernet interface available in the device.

If NDP is disabled, it is possible to add NDP entries in the NDP table to control directly the protocol. This could be useful for security purposes particularly on the public interface of the 6WINDGate.

To enable NDP on an Ethernet interface, the following command has to be entered:

```
6OS{myconfig-ifname}enable ndp
```

To disable NDP on an Ethernet interface, the following command has to be entered:

```
6OS{myconfig-ifname}disable ndp
```

By default, NDP is enabled on all the Ethernet interfaces.




Modifying NDP Entries

If NDP is disabled the following command can be used to modify the Internet-to-Ethernet address translation table entries used by NDP, use the following command:

```
6OS{myconfig-gen}ndpentry host macaddress
```

where:

- *host* has to be specified by an IPv6 address, using Internet dot notation.
- *macaddress*: MAC address is given as six hexa bytes separated by colons.

-
-  **Note** The NDP table is global for the different Ethernet interfaces of the device.
-
-  **Note** The user should avoid to modify the NDP table without disabling the NDP protocol
-
-  **Note** The IPv6 address must be a neighbour address (i.e: share a common prefix with one of the interfaces).
-

To delete NDP entries, use the following command:

```
6OS{myconfig-gen}delete ndpentry host macaddress
```

All the table entries can be deleted using the following command:

```
6OS{myconfig-gen}delete ndpentry all
```

Example:

```
6OS{myconfig-gen}ndpentry 3ABC::7 00:10:5A:64:CB:2B
6OS{myconfig-gen}delete ndpentry 3ABC::7 00:10:5A:64:CB:2B
6OS{myconfig-gen}delete ndpentry all
```

Ethernet interface (eth)

This chapter describes properties and functions specific to physical Ethernet interfaces.

Ethernet Context

Ethernet interface context is invoked with the `ethX_Y` keyword.

```
6OS{myconfig}ethX_Y
```

Where *x* is a card index and *y* a port index on the network card.

Example:

```
6OS{myconfig}eth1_0
6OS{myconfig-eth1_0}
```

Logical Ethernet Bridge group (bnet)

An Ethernet bridge is a set of channels that are bound together, the L2 PDUs (Level 2 Protocol Data Unit) are switched based on their MAC destination address. The channels can be either UDP tunnels, ATM PVCs, physical Ethernet port, VLAN... A set of channels that are bridged is an Ethernet Bridge group.

The local IPv4 and IPv6 stacks send and receive PDUs through a logical interface `bnet i`, representing the Ethernet bridge group.

To create an Ethernet bridge group, use the following command:

```
6OS{myconfig}bnet i
```

where *i* is the identifier of the Ethernet bridge group. The number of bnets is currently limited to 16.

An Ethernet bridge group can be deleted using the following command:

```
6OS{myconfig}delete bnet i
```

Once the Ethernet bridge group is instantiated, it can be used as any Ethernet physical interface.

ATM Bridge

This feature is only available on models equipped with ATM cards.

To add an ATM channel to the group, use the following command:

```
6OS{myconfig-bnet i}bind atm_card vc vp/vc pvc_template "comment"
```

where:

- *atm_card* is the name of the physical ATM interface (see § ATM interface (*atm_ptp*), p. 49),
- *vp/vc* is the VP identifier and the VC identifier of the bound ATM PVC,
- *pvc_template* is the name of the template that describes the traffic contract, the encapsulation...
- *comment* is a short user description of the ATM PVC

Example:

```
6OS{myconfig-bnet0}bind atm3_0 vc 0/100 pvc_template0 "ATM pvc 0/100"
```

This channel can be used to offer the RFC1483 or RFC2684 bridged service.

VLAN Bridge

To add a 802.1q VLAN channel to the group, use the following command:

```
6OS{myconfig-bneti}bind ethernet_card vlan {'tag'|any} ["comment"]
```

where:

- *ethernet_card* is the name of an ethernet interface, this interface can be a physical interface (*eth0_0*) or a logical interface (*bnet*i**),
- *tag* is the value of the 802.1q VLAN tag, it ranges from 1 to 4094, **any** can be used to accept any tagged frame received,
- *comment* is a short user description of the 802.1q VLAN

Example:

```
6OS{myconfig-bnet0}bind eth0_0 vlan 1001 "VLAN 1"
6OS{myconfig-bnet0}bind eth0_0 vlan 1002 "VLAN 2"
6OS{myconfig-bnet1}bind bnet0 vlan 1003 "VLAN 3"
```

This channel can be used to offer 802.1q VLAN service.

Ethernet Raw Bridge

To add a physical Ethernet port to the group, use the following command:

```
6OS{myconfig-bneti}bind ethernet_card raw ["comment"]
```

where:

- *ethernet_card* is the name of an ethernet interface,
- *comment* is a short user description of the raw channel

Example:

```
6OS{myconfig-bnet0}bind eth0_0 raw "physical Ethernet port 0"
```

IP configuration of the Ethernet bridge group

The *bnet*i** logical interfaces are broadcast interfaces like any other physical Ethernet interface.

Example:

```
6OS{myconfig-bnet0}bind atm3_0 vc 0/141 pvc_template1 "pvc to 6WIND"
6OS{myconfig-bnet0}ipaddress 192.168.41.2/24
6OS{myconfig-bnet0}ipaddress 3ffe:305:1017:1::1/64
```

Chapter 8 Configuring loopback interfaces

The loopback interfaces have many purposes. When a loopback address is announced into the routing tables, it can be used as a management address instead of a physical interface address, which is preferable because a loopback interface is independent of any physical interfaces and therefore is always accessible. When configuring unnumbered point-to-point interfaces (for example with a PPPv4 server), you must ensure that at least an address is configured on another interface in the router. This address will be the source address. It is recommended that you do this by assigning an address to a loopback interface. Moreover, when a prefix is set on a loopback interface, it can be used in order to announce some directly connected networks into the routing protocols.

Entering the loopback interfaces context

Loopback interfaces are all grouped in the same context. First enter the `loopback` context:

```
6OS {myconfig} loopback
6OS {myconfig-loopback}
```

Adding a loopback interface

A loopback interface is a special interface with few properties. It is just defined by a unique IPv4 or IPv6 address and prefix length. It cannot be set up or down.

```
6OS {myconfig-loopback} loop index address/prefixlength
```

Example:

```
6OS {myconfig-loopback} loop 1 20.30.40.50/24
6OS {myconfig-loopback} loop 2 20.30.40.55/32
6OS {myconfig-loopback} loop 1 2001:cafe:deca::1/128
```

These commands will create two interfaces, the `loop1` interface with IPv4 address 20.30.40.50 and prefix length 24 and with an IPv6 address, and the `loop2` interface with only an IPv4 address. The `loop1` interface may be used by routing protocols, which will advertise a route to 20.30.40.0/24 network. The `loop1` interface may be used as the router's management address for ssh, snmp or telnet.

Deleting a loopback interface

The delete loop command deletes an existing loopback interface.

```
6OS {myconfig-loopback} delete loop index
```


Example:

```
60S{myconfig-loopback}delete loop 1
```

Chapter 9 Configuring point to point interfaces

ATM interface (atm_ptp)

This chapter describes how to configure an ATM point to point interface. This feature is only available on models equipped with ATM cards.

Configuring an ATM template

To create an ATM point-to-point interface, first create an ATM template. An ATM template can be defined by entering a command made of a “pvc_” prefix followed by a user-specified template name.

```
6OS{myconfig}pvc_templatename
```

where:

- *templatename* is the user-specified name of the ATM template.

Example:

```
6OS{myconfig}pvc_template0
```

The ATM templates are used in order to configure:

- the ATM traffic contract,
- the encapsulation type.

ATM traffic contract (COS)

The ATM traffic contract provided by the 6WINDGate consists of the following five services categories:

- UBR: Unspecified Bit Rate
The Unspecified Bit Rate service category is intended for non-real-time applications, i.e. those not requiring tightly constrained delay and delay variation.
- CBR: Constant Bit Rate
The Constant Bit Rate service category is used by connections that request a static amount of bandwidth that is continuously available during the connection lifetime.
- VBR: Variable Bit Rate
The Variable Bit Rate category is intended for applications which have bursty traffic characteristics and which are characterized in terms of a PCR (Peak Cell Rate), SCR (Sustainable Cell Rate) and MBS (Max Burst Size).

The ATM templates are instantiated with the following command:

```
6OS{myconfig-pvc_template_name}trafficubr [peak cell rate]
```

```
6OS{myconfig-pvc_template_name}traffic cbr [peak cell rate]
```

```
6OS{myconfig-pvc_template_name}traffic vbr [peak cell rate]
[sustainable cell rate] [max burst size]
```

ATM point-to-point PVC (routed mode)

Many AAL5 ATM PVCs can be provisioned on a physical ATM interface. PVCs can support any kind of encapsulations that are described by both the RFC1483 and the RFC2684 and that are listed into the previous section.

The supported routed protocol is IPv4/IPv6 with LLC/SNAP header

A point-to-point ATM interface is instantiated with the following command:

```
6OS{myconfig}atm_ptpi
```

Where *i* is the identifier of the point-to-point ATM interface

A point-to-point ATM interface can be deleted using the following command:

```
6OS{myconfig}delete atm_ptpi
```

Binding the point-to-point interface

To bind the point-to-point interface to an ATM PVC, use the following command:

```
6OS{myconfig-atm_ptpi}bind atm_card vc vp/vc pvc_template "comment"
```

where:

- *atm_card* is the name of the physical ATM interface (see Chapter 10),
- *vp/vc* defines the VP and VC identifiers of the bound ATM PVC,
- *pvc_template* is the name of the template that describes the traffic contract, the encapsulation, ...
- *comment* is a short user description of the ATM PVC

Example:

```
6OS{myconfig-atm_ptp0}bind atm3_0 vc 0/100 pvc_template0 "ATM pvc 0/100"
```

Defining ATM PVC Endpoints

To assign a pair of IPv4 or IPv6 addresses to a PVC ATM interface, use the following command:

```
6OS{myconfig-atm_ptpi}endpoints local@ remote@
```

where:

- *local@* is the IPv4 or IPv6 address of the local endpoint of the PVC,
- *remote@* is the IPv4 or IPv6 address of the remote endpoint of the PVC.

To delete existing endpoints on an ATM PVC, use the following command:

```
6OS{myconfig-atm_ptpi}delete endpoints local@
```

Example:

```
6OS{myconfig-atm_ptp0}endpoints 10.12.37.1 10.12.37.2
```

```
6OS{myconfig-atm_ptp0}endpoints 3abc::1 3abc::2
```

```
6OS{myconfig-atm_ptp0}delete endpoints 10.12.37.1
```

```
6OS{myconfig-atm_ptp0}delete endpoints 3abc::1
```

Chapter 10 *Configuring network card controllers*

Network card controllers are contexts that enable to configure the physical ports of a network interface. These controllers are provided for specific types of network cards, such as multi-protocol serial cards or ATM cards. The controller configures the hardware part of the card, when logical interfaces configure the upper layer protocols.

ATM controller

An ATM controller configures an ATM physical interface. This feature is only available on models equipped with ATM cards.

Entering ATM controller context

The following command enters the ATM controller context:

```
6OS{myconfig}atmX_Y
```

Where *x* is the network card index and *y* a port index on the card.

Example:

```
6OS{myconfig}atm3_0
6OS{myconfig-atm3_0}
```

ATM Signaling

The 6WINDGate supports up to three signaling handlers per interface in order to establish AAL5 PVCs. These modes are set using the following command:

```
6OS{myconfig-atm3_0}signalling {none|uni3.0|uni3.1}
```

The default mode is **none**. It means that no UNI message is exchanged, then only static AAL5 PVCs can be provisioned.

Chapter 11 *Advanced configuration settings*

Configuring Maximum Transmission Unit (MTU)

Presentation

Packet size, often referred to as **MTU** (Maximum Transmission Unit) is the greatest amount of data that can be transferred in one physical frame on the network. For Ethernet, the MTU is 1500 bytes, for PPPoE 1492, ...

The IP MTU size can be adjusted will fragment any IP packet that exceeds the MTU set for an interface.

MTU Configuration Context

The MTU size can be activated on a physical interface such as, Ethernet, and some logical interfaces such as a PPPoE interface.

Configuring MTU

To configure MTU for IPv4 or IPv6 can be done using the following commands:

```
60S{myconfig-interf}mtu {default|value}
```

where:

- *default* is the typical value in bytes for each kind of interface
- *value* is a numeric value between 512 and 65485.

Configuration Example

To configure the value 1200 for **mtu** on the interface **eth2_0**, use the following commands:

```
60S{myconfig}eth2_0
60S{myconfig-eth2_0}mtu 1200
```

Configuring TCP Maximum Segment Size(TCPMSS)

Presentation

TCPMSS is a negotiated value standing for the maximum amount of TCP data in one IP packet. To avoid IP fragmentation, the TCPMSS value must be less than the Path MTU less the TCP/IP header size (for example, with IPv6 over Ethernet $1500 - (40+20) = 1440$; $TCPMSS \leq 1440$).

In order to avoid host Path MTU discovery failure (some firewalls or routers discard the Path MTU protocol), the 6WINDGate can automatically patch this value during the forwarding of TCP negotiation. Indeed, when a TCP packet input or output from any physical interface with the flag SYN, the TCPMSS value is changed in the packet if and only if the new value is less than the old one. By consequence, the TCP connection is set with the smallest TCPMSS value.

The 6WINDGate allows configuring the TCPMSS for IPv6 and IPv4.

For more explanations, please visit our web site, and read the application note “Configuring TCPMSS for DSL connection”

TCPMSS Configuration Context

The TCPMSS adaptation can be activated on a physical interface such as, Ethernet, and some logical interfaces such as a PPPoE interface.

Configuring TCPMSS

To configure TCPMSS for IPv4 or IPv6 can be done using the following commands:

```
60S{myconfig-interf}tcp4mss [value|disable|automatic]
60S{myconfig-interf}tcp6mss [value|disable|automatic]
```

where:

- *value* is a numeric value between 512 and 65485.
- *disable* means that the mechanism is not activated.
- *automatic* means that the new value is computed from the interface MTU, taking in account the IP/TCP overhead (different for IPv4 and IPv6).

Configuration Example

To configure the value 1453 for **tcp4mss** and an *automatic* value for **tcp6mss** on the interface **pppoe2**, use the following commands:

```
60S{myconfig}pppoe2
60S{myconfig-pppoe2}tcp4mss 1453
60S{myconfig-pppoe2}tcp6mss automatic
```

Configuring Maximum Receive Unit (MRU)

This Configuration Option may be sent to inform the peer that the implementation can receive larger packets, or to request that the peer send smaller packets.

For example, if a peer advertises an MRU of 1460, that peer will not process a PPP packet with a payload larger than 1460 bytes long.

```
60S{myconfig-pppoe0}mru {default/value}
```


where:

- **default** is the typical value for each kind of interface.
- *value* is a numeric value between 512 and 65485.

Chapter 12 *Configuring a Serial Interface*

This chapter describes how to configure a serial interface.

Configuring Serial Physical Parameters

Serial Interface Context

All the serial interface parameters have to be defined in the `ser3_0` context using the CLI.

```
6OS{myconfig}ser3_0
```

Binding a Logical Interface to a Physical Interface

Binding a logical interface to a physical interface links a layer 2 protocol to a physical line. It is required only for the serial interface and it can be done using the following command:

```
6OS{myconfig-ser3_0}bind upper_protocol
```

where:

- `upper_protocol` is one of the following protocols `ppp` or `chdlc`

Configuring the Clock

The following command defines the source of the transmit and receive clock signals:

```
6OS{myconfig-ser3_0}clock source
```

where:

- `source` should be set to one of the following source: `external` or `internal`.

The default value is `external`.

Configuring the Line Baud Rate

The following command defines the data transfer rate on a serial interface. This value is meaningful if internal clocking is selected.

```
6OS{myconfig-ser3_0}baudrate baudrate_value
```

where:

- `baudrate_value` should be set to one of the following values: `9.6`, `19.2`, `38.4`, `56`, `64`, `128`, `1544` or `2048`. These values are expressed in Kbps.

The default value is *2048*.

Setting the Interface Up and Down

A serial interface can be set up and down using the following command:

```
6OS{myconfig-ser3_0}intf {up | down}
```

By default the interface is *up*.

Configuration Examples

Example 1: configuring the serial parameters to use PPP protocol

```
6OS{myconfig}ser3_0
# Enter serial interface context.
6OS{myconfig-ser3_0}bind ppp
# Define the protocol to use.
6OS{myconfig-ser3_0}clock internal
# Define source of transmit and receive clock signals.
6OS{myconfig-ser3_0}baudrate 2048
# Define data transfer rate.
6OS{myconfig-ser3_0}exit
```

Example 2: configuring the serial parameters to use CHDLC protocol

```
6OS{myconfig}ser3_0
# Enter serial interface context.
6OS{myconfig-ser3_0}bind chdlc
# Define the protocol to use.
6OS{myconfig-ser3_0}clock internal
# Define source of transmit and receive clock signals.
6OS{myconfig-ser3_0}baudrate 2048
# Define data transfer rate.
6OS{myconfig-ser3_0}exit
```



Configuring Cisco HDLC

CHDLC Context

All the CHDLC parameters have to be defined in the `chdlc0` context using the CLI.

```
6OS{myconfig}chdlc0
```

Configuring CHDLC Parameters

-
-  **Note** Before configuring CHDLC parameters, check the serial parameters configuration and verify that the CHDLC protocol has been defined as the protocol to be used thanks to a `bind chdlc` command.
-
-  **Note** The following parameters (`keepalive_tx_timer`, `keepalive_rx_timer`, `keepalive_err_margin`, `ignore_dcd`, `ignore_cts`, `ignore_keepalive`) cannot be defined using the NMS.
-

Refer to the *6WINDGate SixOS - Command Reference* to obtain details about these parameters and how to modify them. The default values for these parameters are as follow:

- 10000 milliseconds for `keepalive_tx_timer`,
- 10000 milliseconds for `keepalive_rx_timer`,
- 3 milliseconds for `keepalive_err_margin`,
- `yes` for `ignore_dcd`,
- `yes` for `ignore_cts`,
- `yes` for `ignore_keepalive`.

Defining CHDLC Endpoints

IP endpoints have to be configured before using Cisco HDLC.

To assign a pair of IPv4 or IPv6 addresses to a CHDLC interface, use the following command:

```
6OS{myconfig-chdlc0}endpoints local remote
```

- `local` is the IPv4 address of the local endpoint.
- `remote` is the IPv4 address of the remote endpoint.

To delete existing endpoints, use the following command:

```
6OS{myconfig-chdlc0}delete endpoints local
```

Configuration Example

```
6OS{myconfig}chdlc0
# Enter chdlc context.
6OS{myconfig-chdlc0}endpoints 10.17.2.3 10.17.2.4
# Define endpoints.
6OS{myconfig-chdlc0}enable qos in
# Enable QoS.
6OS{myconfig-chdlc0}disable ipsec
# Disable IPsec.
6OS{myconfig-chdlc0}exit
```

Chapter 13 Configuring Point To Point Protocols

This chapter describes how to configure point-to-point protocols such as PPP and PPPoE.

Configuring PPP

PPP Context

All the PPP parameters have to be defined in the `ppp0` context using the CLI.

```
60S{myconfig}ppp0
```

Configuring PPP parameters



Note Before configuring PPP parameters, check the serial parameters configuration and verify that the PPP protocol has been defined as the protocol to be used thanks to a `bind ppp` command.

The following parameters have to be configured before using PPP:

- IP endpoints,
- PAP or CHAP authentication parameters (refer to section Configuring PPP Authentication Protocols).

Defining PPP Endpoints

To assign a pair of IPv4 or IPv6 addresses to a PPP interface, use the following command:

```
60S{myconfig-ppp0}endpoints local remote
```

where:

- *local* is the IPv4 or IPv6 address of the local endpoint.
- *remote* is the IPv4 or IPv6 address of the remote endpoint.

To delete existing endpoints, use the following command:

```
60S{myconfig-ppp0}delete endpoints local
```



Note Defining the address type (IPv4 or IPv6) will automatically activate the corresponding version of PPP (PPPo4 or PPPo6).

Configuring PPPoE

Overview

PPPoE (PPP over Ethernet) is a way for PPP traffic to cross an Ethernet link. All the PPP traffic (Link Control Protocol, Network layer Control Protocol, and authentication) is encapsulated over Ethernet.

This ability can be used by the 6WINDGate on a shared Ethernet link to open PPP sessions to multiple destinations via one or more bridging modems.

PPPoE Context

PPPoE interface can be configured in the `pppoei` context.

```
60S{myconfig}pppoei
```

where:

- `i` indicates the interface number (pppoe0, pppoe1, pppoe2).

Setting the Interface Up and Down

A PPPoE interface can be set up and down using the following command:

```
60S{myconfig-pppoei}intf {up | down}
```

By default the interface is *up*.

Configuring a PPPoE Interface

To configure a PPPoE interface, use the following commands:

```
60S{myconfig-pppoei}bind ethx_0
```

where:

- `ethx_0` is the physical Ethernet interface on which the PPPoE session will be done.

```
60S{myconfig-pppoei}service name
```

where:

- `name` is the service name of your PPPoE session.

The `bind` command is a required command when configuring PPPoE service whereas the `service` command is optional.

Example

```
60S{myconfig-pppoe0}bind eth0_0
60S{myconfig-pppoe1}bind eth0_0
60S{myconfig-pppoe0}service myProvider
```



Note There is no straightforward relation between the index of the `pppoe` interface and the one of the Ethernet interface. For example, it is possible in the `pppoe1` context to provide a PPPoE session over `eth0_0`.

Configuring PPP DNS Extension

PPP DNS extension can be used to configure the IPv4 address of the DNS server for the DNS-proxy function.

Please refer to section “Configuring DNS proxy and Client”.

Configuring PPP Authentication Protocols

Enabling CHAP or PAP

If PPP or PPPoE sessions require authentication, the 6WINDGate supports CHAP or PAP authentication protocols.

To enable or disable CHAP on a link, use respectively the following commands in the `ppp0` or `pppoei` context:

```
60S{myconfig-ppp0}chap enable
60S{myconfig-ppp0}chap disable
60S{myconfig-pppoe0}chap enable
60S{myconfig-pppoe0}chap disable
```

By default, CHAP is disabled.

To enable or disable PAP on a link, use respectively the following commands in the `ppp0` or `pppoei` context:

```
60S{myconfig-ppp0}pap enable
60S{myconfig-ppp0}pap disable
60S{myconfig-pppoe0}pap enable
60S{myconfig-pppoe0}pap disable
```

By default, PAP is disabled.

Defining PAP Parameters

Figure 4 describes how PAP works. The following authentication parameters have to be defined:

- The `authenticator` parameter defines whether your 6WINDGate will be the authenticator or not on a serial interface.
- If the 6WINDGate is the authenticator, a list of all the users that are valid for authentication purposes on a serial interface has to be defined.

The `authenticator` command defines whether your 6WINDGate will be the authenticator or not on a serial interface.

The default value is *no*.

The `userid` command defines a list of all the users that are valid for authentication purposes on a serial interface. This parameter is dependent on the `authenticator` parameter. If `authenticator` is set to:

- *no*, then you will simply enter in your login name and password that the other side specified to you.
- *yes*, then you will have to maintain a list of all the users containing the login name and password that are valid for authentication purposes.

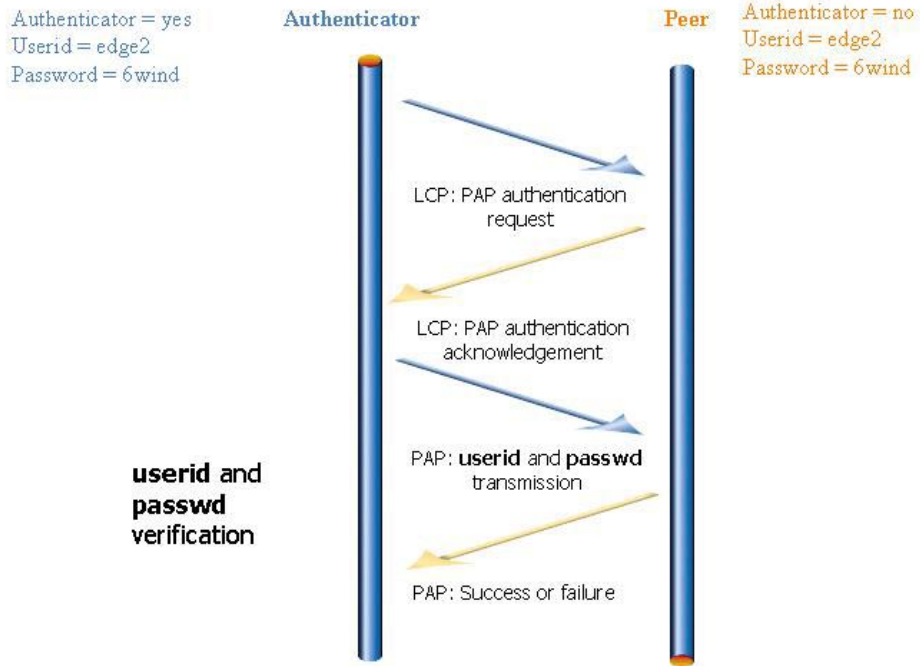


Figure 4: PAP protocol description

Defining CHAP Parameters

Figure 5 describes how CHAP works. The following authentication parameters have to be defined:

- The authenticator parameter defines whether your 6WINDGate will be the authenticator or not on a serial interface.
- If the 6WINDGate is the authenticator, the Challenge System Name has to be defined.
- If the 6WINDGate is the authenticator, a list of all the users that are valid for authentication purposes on a serial interface have to be defined.

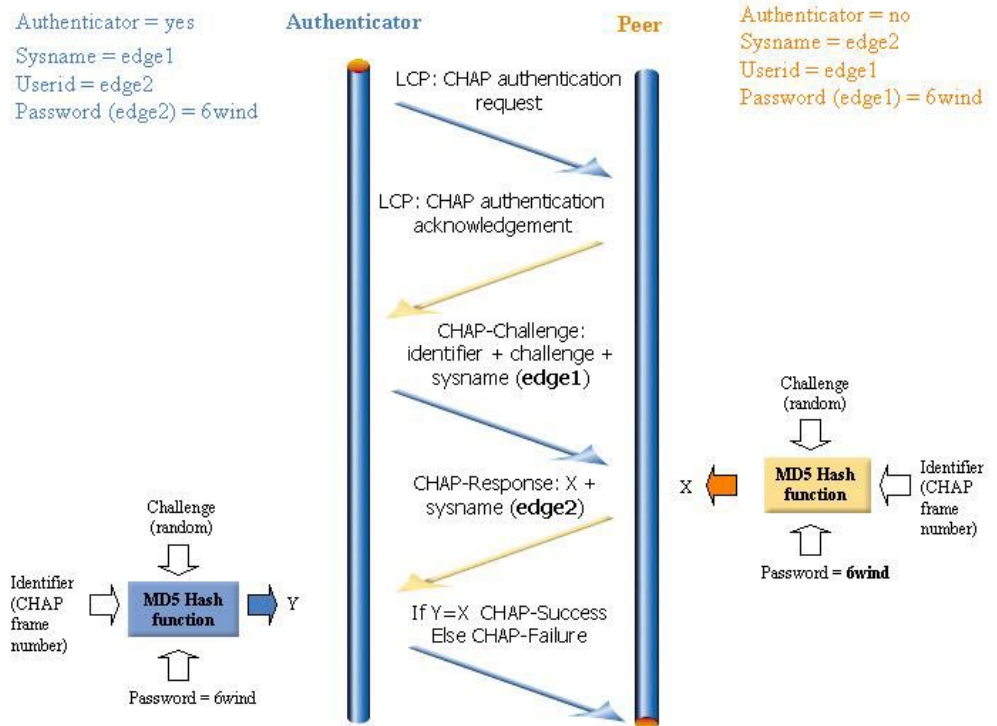


Figure 5: CHAP protocol description

The **authenticator** command defines whether your 6WINDGate will be the authenticator or not on a serial interface.

The default value is *no*.

The **sysname** command defines a Challenge System Name. This parameter is dependent on the **chap** parameter. If **chap** is set to:

- *disable*, then you can simply ignore this parameter.
- *enable*, then you have to enter a name which can be no longer than 31 characters

The **sysname** default value is *hurricane*.

The **userid** command defines a list of all the users that are valid for authentication purposes on a serial interface. This parameter is dependent on the **authenticator** parameter. If

authenticator is set to:

- *no*, then you will simply enter in your login name and password that the other side specified to you.
- *yes*, then you will have to maintain a list of all the users containing the login name and password that are valid for authentication purposes.

Configuration Examples

Example 1: PPPv4 without authentication

```
60S{config6221}ppp0
    # Enter ppp context.
60S{config6221-ppp0}endpoints 10.17.2.3 10.17.2.4
```

```
# Define IPv4 endpoints.
# Don't need a userid when neither PAP nor
# CHAP is enable
60S{config6221-ppp0}exit
```

Example 2: PPPv6 with PAP


```
60S{config6221}ppp0
# Enter ppp context.
60S{config6221-ppp0}endpoints 3abc::1 3abc::2
# Define IPv6 endpoints.
60S{config6221-ppp0}pap enable
# Enable PAP authentication.
60S{config6221-ppp0}userid name password
# Define User ID where name and password must be the same value
# on both devices
# Sysname not required with PAP
60S{config6221-ppp0}authenticator yes
# Define Authenticator.
60S{config6221-ppp0}exit
```

Example 3: PPPv4 with CHAP

```
60S{config6221}ppp0
# Enter ppp context.
60S{config6221-ppp0}endpoints 10.17.2.3 10.17.2.4
# Define IPv4 endpoints.
60S{config6221-ppp0}chap enable
# Enable CHAP authentication.
60S{config6221-ppp0}userid name password
# Define User ID where name refer to the name of
# the remote device.
60S{config6221-ppp0}sysname name
# Define Sysname where name refer to the name of the
# local device.
60S{config6221-ppp0}authenticator yes
# Define Authenticator.
60S{config6221-ppp0}exit
```

Example4: PPPoE with PAP

```
60S{myconfig}pppoe0
60S{myconfig-pppoe0}bind eth1_0
60S{myconfig-pppoe0}authenticator no
60S{myconfig-pppoe0}pap enable
60S{myconfig-pppoe0}userid fti/yptUI08@fti 6winDontheWind
# login and password for the PPPoE connection
60S{myconfig-pppoe0}sysname fti/yptUI08@fti
# the sysname refers to the userid to use for the PPPoE
# connection
```

 **Note** The `sysname` command has a different meaning in PPPoE. This command must refer to one of the `userid` you are using for authentication.

Example2: PPPoE with CHAP

```
60S{myconfig}pppoe0
60S{myconfig-pppoe0}bind eth1_0
60S{myconfig-pppoe0}authenticator no
60S{myconfig-pppoe0}chap enable
60S{myconfig-pppoe0}userid fti/yptUI08@fti 6winDontheWind
                        # login and password for the PPPoE connection
60S{myconfig-pppoe0}sysname fti/yptUI08@fti
                        # the sysname refers to the userid to use for the PPPoE
                        connection
```

Rebooting the 6WINDGate 6200 series



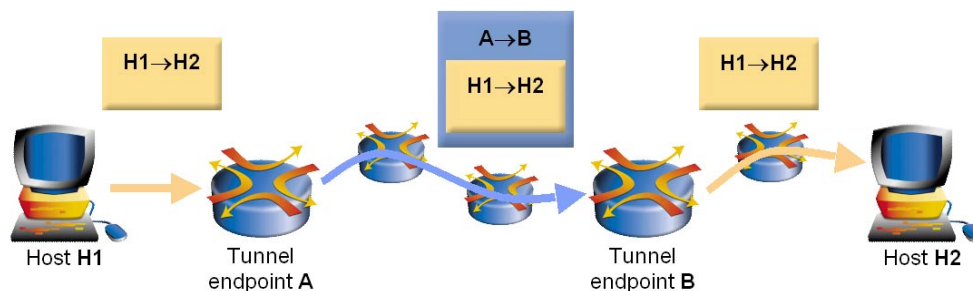
Note `apply` and `addrunning` commands do not process `ser3_0`, `ppp0` and `chd1c0` context parameters for 6WINDGate 6200 series. The user has to make the configuration bootable and to reboot the 6WINDGate to configure the serial interface with the new parameters.

Chapter 14 Configuring IPv4 and IPv6 Tunneling

This chapter describes how to configure IPv4 and IPv6 tunneling techniques.

Introduction

Tunneling is a widespread technique used in networking, in order to resolve many problems: IPv4/IPv6 migration, Virtual Private Networks, routing. It consists in encapsulating a layer 3 PDU (Protocol Data Unit), typically an packet, into a new layer 3 packet, by appending an IP header. The 6WINDGate provides several techniques to tunnel IP packets into new IP packets (the inner and outer IP versions may differ).



Tunneling techniques create a virtual layer 2 link (called a tunnel) between the source and destination of the encapsulating packets, and hide the network topology between these two endpoints, as if the two endpoint were directly connected. Therefore, the 6WINDGate creates a logical point-to-point interface, that appears in the list of interfaces and that can be used by other functions, notably routing.

Definitions

Tunneling

Tunneling provides an essential IPv6 migration mechanism. It also allows to simulate VLANs.

A very likely migration scenario is that some IPv6 networks will appear at the periphery of the IPv4 Internet. Tunneling techniques make it possible to connect these IPv6 clouds without any IPv6 native connection.

The IPv6 basic specifications introduce three types of tunnels:

- Automatic tunnels
- Configured tunnels
 - o 6in4,

- o 4in6,
- o 6in6.
- 6to4 tunnels

As detailed in Figure 1, tunnels consist in encapsulating IP packets into other IP packets. These IP packets are then transmitted across the network to the tunnel end-point, where the original IP packets are extracted from the encapsulating packets

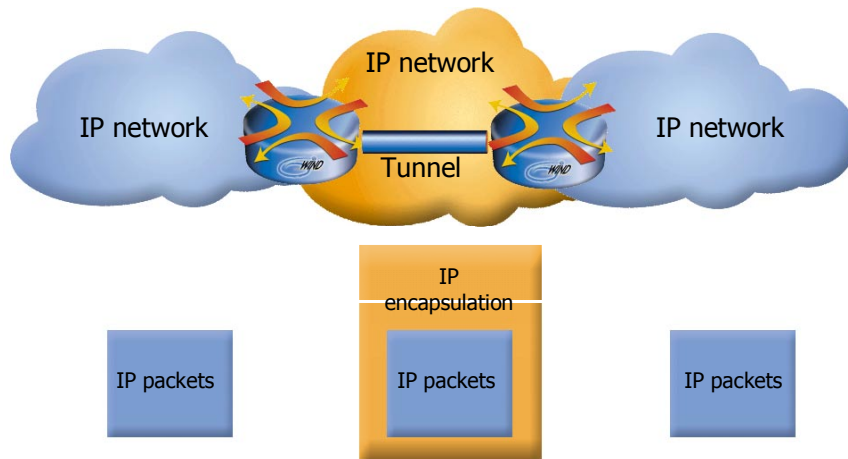


Figure 6: Tunneling – IP encapsulation

Automatic Tunnel

Automatic tunnels use “IPv4-compatible” addresses, which are hybrid IPv4/IPv6 addresses. A compatible address is created by adding leading zeros to a 32-bit IPv4 address to pad it out to 128 bits.

Automatic tunnels are not associated to any remote end point. As a consequence, automatic tunneling can be attached to several IPv4 addresses, themselves attached to physical interfaces. So, there is a need for only one automatic tunnel.

IPv6 in IPv4 Configured Tunnel

An IPv6 in IPv4 configured tunnel encapsulates IPv6 traffic in an explicit IPv4 tunnel. Such a tunnel is an association of two tunnel end points. For each end point, an IPv4 and an IPv6 address have to be configured.

The number of IPv6 in IPv4 configured tunnels end-points is currently limited to 32.

IPv4 in IPv6 Configured Tunnel

IPv4 in IPv6 Configured Tunnels could be useful for the interconnection of IPv4 clouds on an IPv6 native service.

An IPv4 in IPv6 configured tunnel encapsulates IPv4 traffic in an explicit IPv6 tunnel. Such a tunnel is an association of two tunnel end points. For each end point, an IPv4 and an IPv6 address have to be configured.

The number of IPv4 in IPv6 configured tunnels end-point is currently limited to 32.

IPv6 in IPv6 Configured Tunnel

IPv6 in IPv6 Configured Tunnels could be useful to simulate VLANs.

An IPv6 in IPv6 configured tunnel encapsulates IPv6 traffic in an explicit IPv6 tunnel. Such a tunnel is an association of two tunnel end points. For each end point, an encapsulated IPv6 and an encapsulating IPv6 addresses have to be configured.

The number of IPv6 in IPv6 configured tunnels end-point is currently limited to 32.

6to4

The “6to4” transition mechanism, “Connection of IPv6 Domains via IPv4 Clouds without Explicit Tunnels”, provides a solution to the problem of using manually configured tunnels by specifying a unique routing prefix for each end-user site that carries an IPv4 tunnel endpoint address.

6to4 tunnels are not associated to any remote end point. As a consequence, the 6to4 tunneling can be attached to several IPv4 addresses, themselves attached to physical interfaces. So, there is a need for only one automatic 6to4 tunnel.

ISATAP

The ISATAP (Intra Site Automatic Tunnel Addressing Protocol) provides a solution to the problem of a host with IPv4/IPv6 dual stack isolated in a IPv4 cloud that tries to establish an IPv6 connection to a gateway with IPv4/IPv6 dual stack. The connection is provided through an automatically configured IPv6 in IPv4 tunnel

Migration Techniques

All the commands related to the IPv4/IPv6 migration configuration are configured in the **mig** context.

Entering the Migration context

To enter this context, just type the **mig** keyword from a configuration context or from a function context.

```
6OS{myconfig}mig
6OS{myconfig-mig}
```

Displaying Migration Configuration

To display information about IPv4/IPv6 migration functions, the following commands can be used:

```
6OS{myconfig-mig}display
6OS{myconfig}display mig
```

Automatic Tunnel Configuration

Creating a Local Automatic Tunnel End-Point

To define a local automatic tunnel endpoint, use the following command:

```
6OS{myconfig-mig}automatic ipv4address
```

ipv4address is the IPv4 address to be used as the local tunnel endpoint. It must be a valid address on one of the interfaces.

Example

```
6OS{myconfig-mig}automatic 192.0.0.1          #::192.0.0.1 IPv6 address
                                                # of the local endpoint of
                                                # the tunnel
```

This command will configure the logical interface named tu0 (for automatic Tunnel) with the “compatible IPv6 address” ::192.0.0.1.

Deleting the Address of the Automatic Tunnel End-Point

To delete a local automatic tunnel endpoint, use the following command:

```
6OS{myconfig-mig}delete automatic ipv4address
```

where:

- *ipv4address* is the IPv4 address to be deleted as the local tunnel endpoint.

Example:

```
6OS{myconfig-mig}delete automatic 192.0.0.1
```

IPv6 in IPv4 Configured Tunnel Configuration

An IPv6 in IPv4 Configured Tunnel is the association of two tunnel end-points. The equipments at each tunnel end point must be configured in a coherent manner.

Creating an IPv6 in IPv4 Configured Tunnel

To define an IPv6 in IPv4 configured tunnel, use the following command:

```
6OS{myconfig-mig}6in4 number encapsulating_localv4 encapsulating_remotev4 localv6
remotev6
```

where:

- *number* is the number (between 0 and 31) identifying the name of the IPv6 in IPv4 configured tunnel interface. The name identifies the tunnel locally.
- *encapsulating_localv4* is the IPv4 address of the local endpoint of the tunnel. It must be a valid address on one of the interfaces.
- *encapsulating_distantv4* is the IPv4 address of the remote endpoint of the tunnel. It must be a valid address on one of the interfaces of the remote end point.
- *localv6* is the IPv6 address of the local endpoint of the tunnel.
- *distantv6* is the IPv6 address of the remote endpoint of the tunnel.

Example:

```
6OS{myconfig-mig}6in4 1 192.0.0.1 192.0.0.2 3ffe::1 3ffe::2
                                                # define 6in4 number 1
```

This command will create a logical interface named ctu1 (for Configured Tunnel) and whose global IPv6 address is 3ffe::1.

Deleting an IPv6 in IPv4 configured tunnel

To delete the address of the IPv6 in IPv4 configured tunnel end-point, use the following command:

```
6OS{myconfig-mig}delete 6in4 number
```

where:

- *number* is the number (between 0 and 31) identifying the IPv6 in IPv4 configured tunnel interface.

Example:

```
6OS{myconfig-mig}delete 6in4 2 # delete 6in4 number2
```

IPv4 in IPv6 Configured Tunnel Configuration

An IPv4 in IPv6 Configured Tunnel is the association of two tunnel end-points. The equipments at each tunnel end point must be configured in a coherent manner.

Creating an IPv4 in IPv6 Configured Tunnel

To create an IPv4 in IPv6 configured tunnel, use the following command:

```
6OS{myconfig-mig}4in6 number encapsulating_localv6 encapsulating_remotev6 localv4
remotev4
```

where:

- *number* is the number (between 0 and 31) identifying the name of the IPv6 in IPv4 configured tunnel interface. The name identifies the tunnel locally.
- *localv6* is the IPv6 address of the local endpoint of the tunnel. It must be a valid IPv6 address on one of the interfaces.
- *remotev6* is the IPv6 address of the remote endpoint of the tunnel. It must be a valid IPv6 address on one of the interfaces of the remote end point.
- *localv4* is the IPv4 address of the local endpoint of the tunnel.
- *remotev4* is the IPv4 address of the remote endpoint of the tunnel.

Example:

```
6OS{myconfig-mig}4in6 1 3ffe::1 3ffe::2 192.0.0.1. 192.0.0.2
# define 4in6 number1
```

This command will create a point-to-point logical interface named `stu1` (for Six Tunnel) and whose global IPv4 address is 192.0.0.1.

Deleting an IPv6 in IPv4 Configured Tunnel

To delete the address of the IPv4 in IPv6 tunnel end-point, use the following command:

```
6OS{myconfig-mig}delete 4in6 number
```

where:

- *number* is the number (between 0 and 31) identifying the name of the IPv4 in IPv6 configured tunnel interface.

Example:

```
6OS{myconfig-mig}delete 4in6 2 # delete 4in6 number 2
```

ISATAP Configuration

To configure an ISATAP connection, first define an ISATAP logical-interface, then declare the IPv6 prefixes owned by that logical-interface.

Activating the ISATAP Mechanism

The ISATAP mechanism is activated as soon as there is one defined well-known address on a logical-interface. To define a well-known address, two steps are required: first configure the ISATAP logical-interface; secondly configure an ISATAP prefix on the same logical-interface.

Configuring an ISATAP logical-interface

To configure an ISATAP logical-interface, use the following command:

```
6OS{myconfig-mig}isatap_router number ipv4address
```

where:

- *number* is the ID number of the ISATAP logical-interface.
- *ipv4address* is the IPv4 address to be used to build a well-known ISATAP IPv6 address *IsatapPrefix::5efe:Ipv4address*.

Example:

```
6OS{myconfig-mig}isatap_router 1 192.0.0.1 # The ISATAP well-known
# address will be:
# IsatapPrefix::5EFE:C000:1
```

This command will create a point-to-point logical interface named `isatap1`.

Configuring an ISATAP prefix

To configure an ISATAP prefix on a specific logical-interface, use the following command:

```
6OS{myconfig-mig}isatap_prefix number IsatapPrefix/Len
```

where:

- *number* is the ID number of the ISATAP logical-interface.
- *IsatapPrefix* is the IPv6 prefix to be used to build a well-known ISATAP IPv6 address *IsatapPrefix::5efe:Ipv4address*.
- *Len* is the length of the *IsatapPrefix*.

Example:

```
6OS{myconfig-mig}isatap_prefix 1 256d:a3b6::/64
# The ISATAP well-known address will be:
# 256D:A3B6::5EFE:C000:1 /64
```

Deleting an ISATAP prefix

To delete an ISATAP prefix, use the following command:

```
6OS{myconfig-mig}delete isatap_prefix number {Prefix/Len|all}
```

where:

- *Number* is the ID number of the ISATAP logical-interface.
- *IsatapPrefix* is the IPv6 prefix to be used to build a well-known ISATAP IPv6 address *IsatapPrefix::5efe:Ipv4address*.
- *Len* is the length of the *IsatapPrefix*.

Example:

```
6OS{myconfig-mig}delete isatap_prefix 0 all
6OS{myconfig-mig}delete isatap_prefix 1 256d:a3b6:: /64
```

Deleting an ISATAP logical-interface

To delete an ISATAP logical-interface, make sure that any prefix be present on this logical-interface yet and then use the following command:

```
6OS{myconfig-mig}delete isatap_router number
```

where:

- *Number* is the ID number of the ISATAP logical-interface.

Example:

```
6OS{myconfig-mig}delete isatap_router 0
```

Deleting the Address of the Tunnel End-Point

To delete the address of the IPv4 in IPv6 tunnel end-point, use the following command:

```
6OS{myconfig-mig}delete 4in6 number
```

where:

- *number* is the number (between 0 and 31) identifying the name of the IPv4 in IPv6 configured tunnel interface.

Example:

```
6OS{myconfig-mig}delete 4in6 2 # delete 4in6 number 2
```



Caution The pseudo-interface for 4in6 and 6in6 tunnels is the same. Thus, if one 4in6 and one 6in6 tunnels are defined on the same pseudo-interface, the encapsulating addresses must be the same.

Activating the 6to4 Mechanism

The 6to4 gateway mechanism is activated as soon as there is one defined well-known address.

Defining a Well-Known 6to4 Address

To define a well-known 6to4 IPv6 address for the 6to4 gateway, use the following command:

```
6OS{myconfig-mig}6to4 ipv4address
```

where:

- *ipv4addr* is the IPv4 address to be used to build a well-known 6to4 IPv6 address
`2002:ipv4address::ipv4address.`

Example:

```
6OS{myconfig-mig}6to4 192.0.0.1 # The 6to4 well-known address will
# be 2002:C000:1::C000:1
```

This command will create a point-to-point logical interface named `stf0` (for Six To Four) with IPv6 addresses `2002:c000:1::c000:1/128` and `inet6 2002::1/16`.

Deleting a Well-Known 6to4 Address

To delete the address of the tunnel end-point, use the following command:

```
6OS{myconfig-mig}delete 6to4 ipv4address
```

The 6to4 address derived from *ipv4Address* (`2002:ipv4address::ipv4address`) is deleted.

Example:

```
6OS{myconfig-mig}delete 6to4 192.0.0.1
```

IPv6 in IPv6 tunnels

Although an IPv6 in IPv6 tunnel is not an IPv4 to IPv6 migration mechanism, it uses the same principles as 6in4 and 4in6 tunnels, and hence is also configured into the `mig` context.

IPv6 in IPv6 Configured Tunnel Configuration

An IPv6 in IPv6 Configured Tunnel is the association of two tunnel end-points. The equipments at each tunnel end point must be configured in a coherent manner.

Creating an IPv6 in IPv6 Configured Tunnel

To create an IPv6 in IPv6 configured tunnel, use the following command:

```
6OS{myconfig-mig}6in6 number encapsulating_localv6 encapsulating_remotev6 localv6
remotev6
```

where:

- *number* is the number (between 0 and 31) identifying the name of the IPv6 in IPv4 configured tunnel interface. The name identifies the tunnel locally.
- *encapsulating_localv6* is the IPv6 encapsulating address of the local endpoint of the tunnel. It must be a valid IPv6 address on one of the interfaces.
- *encapsulating_remotev6* is the IPv6 encapsulating address of the remote endpoint of the tunnel. It must be a valid IPv6 address on one of the interfaces of the remote end point.
- *localv6* is the IPv6 encapsulated address of the local endpoint of the tunnel.
- *remotev6* is the IPv4 encapsulated address of the remote endpoint of the tunnel.

Example:

```
6OS{myconfig-mig}6in6 1 3ffe:34:107::1 3ffe:34:107:1::2 8888::1 8888::2
# define 6in6 tunnel number 1
```

This command will create a point-to-point logical interface named `stu1` (for Six Tunnel) and whose global IPv6 address is `3ffe:34:107::1`.

DSTM

Goal of DSTM

Dual Stack Transition Mechanism (DSTM) addresses the problem of IPv6 hosts in native IPv6 networks which need to maintain connectivity with IPv4-only hosts. DSTM provides a method that ensures this connectivity, based on the use of IPv4-in-IPv6 tunnels and the temporary allocation of a global IPv4 address to hosts requiring such communication.

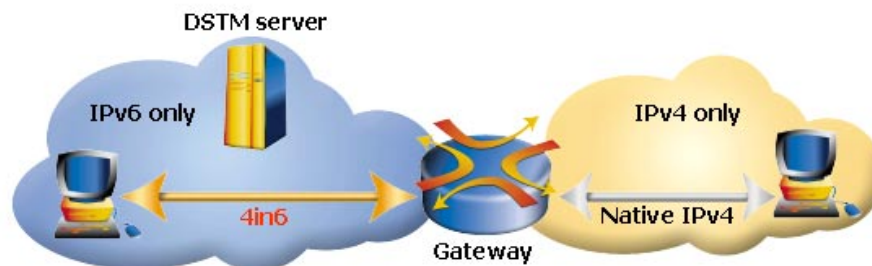


Figure 7: Dual Stack Transition Mechanism

DSTM process

DSTM uses three different types of elements:

□ The address server is in charge of IPv4 address allocation to client nodes. The DSTM server guarantees the uniqueness of the IPv4 address it allocates. The allocation is performed for a limited period of time.

□ The gateway, or Tunnel End Point (TEP), is a border router between the IPv6-only domain and an IPv4 Internet or intranet. This node performs encapsulation and decapsulation of packets. It implements the bi-directional forwarding between both networks. The TEP and the address server may be collocated in the same device.

DSTM nodes are dual-stack hosts located in the IPv6-only domain. They dynamically configure their IPv4 stack by asking the server a temporary IPv4 address and then set up 4in6 tunnels towards a Tunnel End Point.

The following figure details the DSTM process:

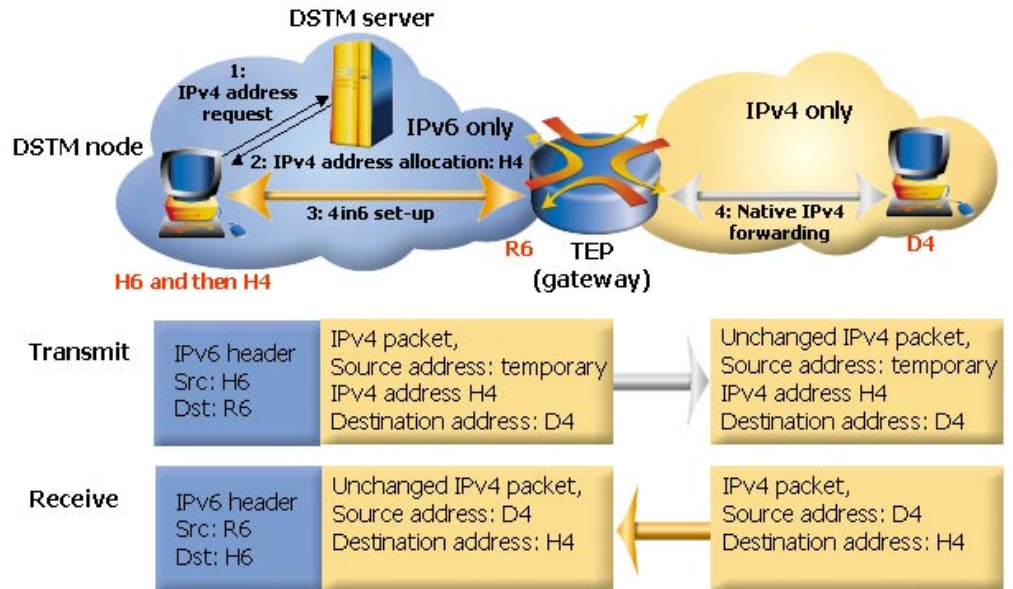


Figure 8: DSTM process

When a DSTM node wishes to communicate with an IPv4 node located elsewhere, it first requests the allocation of a temporary IPv4 address to the DSTM server. This allocation can be performed by several means: DHCPv6, RPC, etc.

Once the temporary address has been received, the DSTM node shall automatically set it up and shall set a 4in6 tunnel to the TEP. The DSTM node is then able to initiate communications with its IPv4 peers:

- An IPv4 application is used on the DSTM node,
- It generates IPv4 packets,
- Packets are transmitted to the TEP through the 4in6 tunnel.

When receiving the packets, the TEP decapsulates them and forward the original IPv4 packets to their destinations. When destinations answer, the TEP encapsulates the IPv4 packets and forwards them back to the adequate IPv6 source.

In this process, IPv4 packets are not necessarily altered. However, the TEP may be associated with an IPv4 NAT. In this case, decapsulated IPv4 packets are submitted to the NAT which alters them.

DSTM applicability

One of the main advantages of DSTM is that it enables a host located in an IPv6-only network to communicate with another IPv4-only host. Moreover, this mechanism does not necessarily alter IPv4 packets on the fly whereas a translation mechanism does. As a consequence, end to end protocols such as IPsec or Voice over IP are not always broken by DSTM.

However, it must be noted that DSTM has not yet been fully accepted at the IETF. Consequently, it has not been widely implemented, especially in hosts.

Configuring DSTM

The 6WINDGate can stand as DSTM router, acting as the formerly described gateway or TEP.

To activate DSTM routing for a subnet, use the following command:

```
6OS{myconfig-mig}dstm interface
```

where:

- `interface` is the interface of the gateway which is connected to the subnet the hosts belong to.

Example:

```
6OS{myconfig-mig}dstm eth0_0
```

This command will activate DSTM routing for hosts belonging to the subnet connected to `eth0_0` interface.

Chapter 15 Auto-configuration functions

Introduction – stateful vs. stateless auto-configuration

Static configuration of IP-capable equipments has proved to be a pain in the neck for network administrators, when they manage a large number of PCs, printers, routers and so on, but also for access or service providers, who manage a large number of subscribers.

Two main auto-configuration protocol families have been developed:

- statefull auto-configuration mechanisms
- stateless auto-configuration mechanisms

Statefull mechanisms make use of servers who register information about all client equipments. Client equipments connect to these servers, possibly authenticate themselves, and request configuration parameters, such as client addresses, client domain name, name server address... The main advantages of these mechanisms is the possibility to do AAA, to deliver numerous and various parameters at the same time and adapt parameters the client identity. The disadvantage os these solutions is the need for centralized servers, or of potentially complex server relay systems. Most common stateful mechanisms are DHCP (Dynamic Host Configuration Protocol) and DHCPv6 (DHCP for IPv6).

Stateless mechanisms are more lightweight mechanisms, where some privileged equipments (typically routers) regularly advertise information common to all hosts on a link, typically network prefixes, default router address, addresses of servers and so on. Hosts on the link receive this information, possibly combine it to specific data (such as network cards MAC addresses or random numbers) and auto-configure themselves, without the need to explicitly ask a server. Hosts are responsible for checking that their parameters do not conflict with those of neighbors' and for notifying servers if necessary (e.g. update Domain Name Servers). These technics alleviate the load of devices composing the network infrastructure, but reduce control of the administrator on the hosts. Notably, AAA is not possible and parameters are the same for all hosts. Moreover, fewer parameters are usually advertised by these mechanisms. The only stateless mechanism is currently "IPv6 stateless auto-configuration".

IPv6 Stateless Auto-configuration

Overview

IPv6 specification describes the stateless address configuration as a possible way to configure IPv6 addresses. This method relies on the IPv6 address structure. IPv6 addresses are made of a network prefix and an interface identifier. Networks prefixes are advertised on every link by routers while the interface identifier is built locally in the host from the MAC address of the network card. From these elements, every host can build its own IPv6 addresses.

The configuration is limited to prefix configuration in routers, since host machines automatically configure themselves.

A 6WINDGate may play different roles:

- When configured as a router, the 6WINDGate periodically advertises prefixes configured on its interfaces. In general, the router does not listen to prefixes advertised by other routers.
- When configured as an auto-configurable device, the listens to prefix advertisements to build its IPv6 addresses.
- When configured as a non auto-configurable device, the 6WINDGate ignore prefix advertisements.

IPv6 Auto-configuration for a Router

When a 6WINDGate is configured as a router, it advertises prefixes based on the Router Advertisement parameters.

Adding an IPv6 prefix on an interface can be done using the following command:

```
6OS{myconfig-ifname}prefix prefix/prefixlen
```

Removing a prefix is done using the following command:

```
6OS{myconfig-ifname}delete prefix prefix/prefixlen
```

Defined prefixes can be displayed using the following command:

```
6OS{myconfig-ifname}display prefix
```

Example

```
6OS{myconfig-eth1_0}prefix 3ffe:304:124:1950::/64
6OS{myconfig-eth1_0}prefix cafe:deca:124:1950::/64
6OS{myconfig-eth1_0}delete prefix cafe:deca:124:1950::/64
```

In general, the router does not listen to any prefix advertised by other routers. By default, auto-configuration is disabled on all the interfaces.

IPv6 prefix are advertised in a RA (Router Advertisement) message. All the parameters relevant to a RA message can be configured using the following command.

```
6OS{myconfig-ifname}router_advert {always|never|smart} interval lifetime
{none|address|other|full} mtu {yes|no}
```

where:

- **always** | **never** | **smart** is the RA transmit mode.
 - o **always** specifies that RA messages are systematically sent, at the period set by *interval* value.
 - o **never** specifies that RA messages are never sent.
 - o **smart** specifies that RA messages are sent only if there are prefixes to advertise. They are sent periodically and in response to a router solicitation message.
- *interval* is the period (in milliseconds) of emission of RA messages.
- *lifetime* is the lifetime (in seconds) of RA messages.
- **none** | **address** | **other** | **full** specifies flags that will appear in RA messages.
 - o **none** specifies that nothing to autoconfigure.
 - o **address** specifies that the address should be configured by DHCP.

- o **other** specifies that some other parameter should be configured by DHCP.
- o **full** specifies that most parameters should be configured by DHCP.
- *mtu* is the MTU value advertised in RA messages. If the value is 0, the MTU is not advertised.
- **yes|no** tells if the *interval* value should be advertised in the router advert message.

The default values for this command are: `smart 30000 1800 none 0 yes`

Example

```
6OS{myconfig-eth1_0}router_advert smart 10000 1800 none 0 yes
```

IPv6 Auto-configuration for an Auto-configurable Device

When a 6WINDGate is configured as an auto-configurable device, the IPv6 stateless address auto-configuration process can be enabled on a broadcast interface using the following command:

```
6OS{myconfig-ifname}enable autoconfv6
```

Once the IPv6 stateless address auto-configuration is enabled, the 6WINDGate is ready to listen prefixes advertised by routers to build its own addresses on the interface. The device does not advertise any prefix.

IPv6 Auto-configuration for a Non Auto-configurable Device

When a 6WIND equipment is configured as a non auto-configurable device, the IPv6 stateless address auto-configuration process can be disabled on an Ethernet interface using the following command.

```
6OS{myconfig-ifname}disable autoconfv6
```

Once the IPv6 stateless address auto-configuration is disabled, the 6WINDGate will not listen to prefixes advertised by routers and will not advertise any prefix.

Chapter 16 Configuring DHCPv6 Client and Prefix Delegation

This chapter describes how to configure the DHCPv6 client.

Introduction

DHCPv6 has been defined for IPv6 knowing that the DHCPv6 protocol is much more simple for IPv6 which integrates a stateless autoconfiguration mechanism (refer RFC 2462).

The 6WINDGate implements a DHCPv6 client to get dynamically IPv6 DNS addresses and delegated prefix from a DHCPv6 server.

The delegating router will be typically present at the point of presence in ISP network, and does not require knowledge about the topology of the links attached to the CPEs. A basic use case is assignment a prefix /48 to CPE that assign a subnet /64 from this delegated prefix to its LAN interfaces, and begin sending router advertisements.

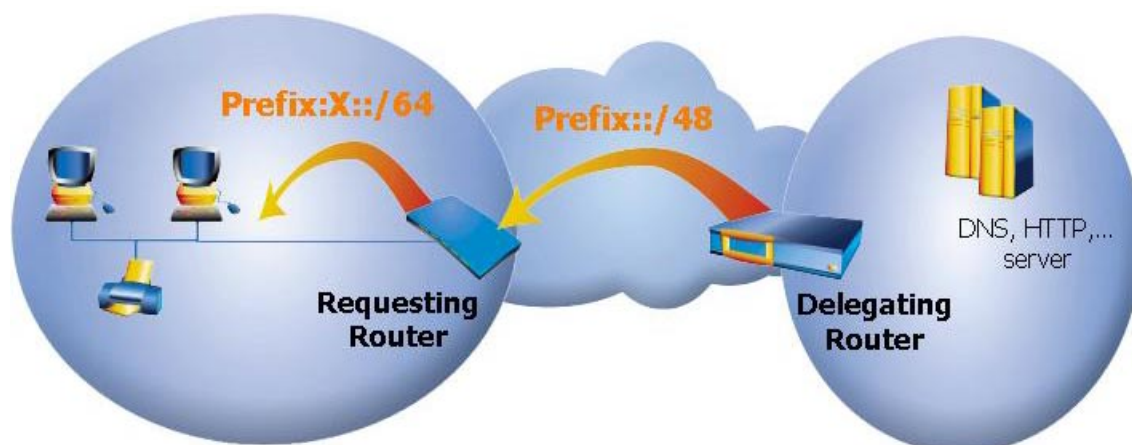


Figure 9. Use case of Prefix delegation

The DHCPv6 client complies with the following IETF drafts:

- draft-ietf-dhc-dhcpv6-28 (Dynamic Host Configuration Protocol for IPv6).
- draft-ietf-dhc-dhcpv6-opt-prefix-delegation-02 (IPv6 Prefix Options for DHCPv6).
- draft-ietf-dhc-dhcpv6-opt-dnsconfig-02 (DNS Configuration options for DHCPv6).

DHCPv6 Client Context

The DHCPv6 client parameters can be configured in the **dhcpv6** context.

```
6OS{myconfig}dhcpv6
```

Displaying DHCPv6 client Configuration

The DHCPv6 client configuration can be displayed using the following commands:

```
6OS{myconfig}display dhcpv6
```

or

```
6OS{myconfig-dhcpv6}display
```

Showing DHCPv6 Client Information

Information learnt through DHCPv6 can be displayed using with the following command:

```
6OS{}show dhcpv6
```

DNS servers addresses learnt by DHCPv6 are listed with the following command:

```
6OS{}show dns
```

A reject route is automatically added by DHCPv6. It can be viewed in the routing table using with the following command:

```
6OS{show-route}show ipv6 route
```

Example

```
6OS{}show dns
# STATIC DNS
# DYNAMIC V4 DNS
  10.130.0.72
# DYNAMIC V6 DNS
  2001:7a8:1090:1300::72
  3ffe:1::73

6OS{show-route}show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng, O - OSPFv3,
       B - BGP, D - DEP, * - FIB route.

S>* ::/0 [1/0] is directly connected, pppoe0
C>* ::/96 is directly connected, tu0
C>* ::1/128 is directly connected, lo0
D>* 3ffe:2::/48 [10/0] via ::1, lo0, rej, 00:37:26
C>* 3ffe:2::/64 is directly connected, eth0_0
C * fe80::/64 is directly connected, pppoe0
C * fe80::/64 is directly connected, eth0_0
C>* fe80::/64 is directly connected, lo0
C>* fec0:0:0:ffff::1/128 is directly connected, eth0_0

6OS{}show dhcpv6
# LEARNT DNS
  3ffe:111:2::2
  3ffe:2222:333:444:555f:6666:ffff:fff1
# PREFIX
  (STATIC)
  (DHCP)    3ffe:2:5::/64
# DUID
```

```
DUID: 00:02:00:00:1c:a8:30:33:30:31:30:30:30:33
```

Enabling and Disabling DHCPv6 Client

The DHCPv6 client can be enabled and disabled using the following commands:

```
6OS{myconfig-dhcpv6}enable dhcpv6
6OS{myconfig-dhcpv6}disable dhcpv6
```

Defining Interface

To define on which the DHCPv6 server can be reached, use the following command:

```
6OS{myconfig-dhcpv6}interface interface
```

When using DHCPv6 for prefix delegation, the interface is generally the external interface connected to ISP domain (WAN interface).

Prefix Delegation Configuration

Prefix Delegation request

To ask a remote DHCPv6 server for a delegated prefix, the following command can be used:

```
6OS{myconfig-dhcpv6}enable prefix_delegation_request
6OS{myconfig-dhcpv6}disable prefix_delegation_request
```

When a prefix P::/n is received, and prefix length n is lower than 64 (typically 48 bits), an identifier (SLA part of IPv6 address) must be given in order to complete the delegated prefix to a 64 bit prefix.

Service Level agreement ID

An identifier must be defined on each interface where you wish to advertise 64 bits prefixes (for stateless auto-configuration). It can be done using the following command:

```
6OS{myconfig-dhcpv6}sla_id interface value
```

where:

- *interface* on which you will the prefix got with the DHCPv6 server .
- *value* corresponds to the 16 last bits of the prefix

For example, the following command adverts the prefix P::/64 (leading zeroes are appended to the delegated prefix up to the 64th bit, starting from most significant bit).

```
6OS{myconfig-dhcpv6}sla_id eth0_0 0x0
```

DUID

The **DUID** is a **DHCPv6 Unique Identifier** of the 6WINDGate. The **DUID** consists in a two octets type code represented in network order, followed by a variable number of octets. The **DUID** can be no more than 256 octets long (not including the type code).

You can choose **Automatic DUID**, then it will be generated by the 6WINDGate. In this case, the **DUID** will be build using the 6WIND enterprise number and the 6WINDGate serial number.

```
6OS{myconfig-dhcpv6}duid auto
```

If you can also enter a static DUID, then you have to enter the **DUID** given by your network provider. Octets must be coded in hexadecimal digit.

```
6OS{myconfig-dhcpv6}duid value
```

Prefix Delegation Negotiation

The negotiation between the DHCPv6 server and client can be making in two or four messages. The basic scheme is described on the Figure 10. DHCPv6 negotiation.

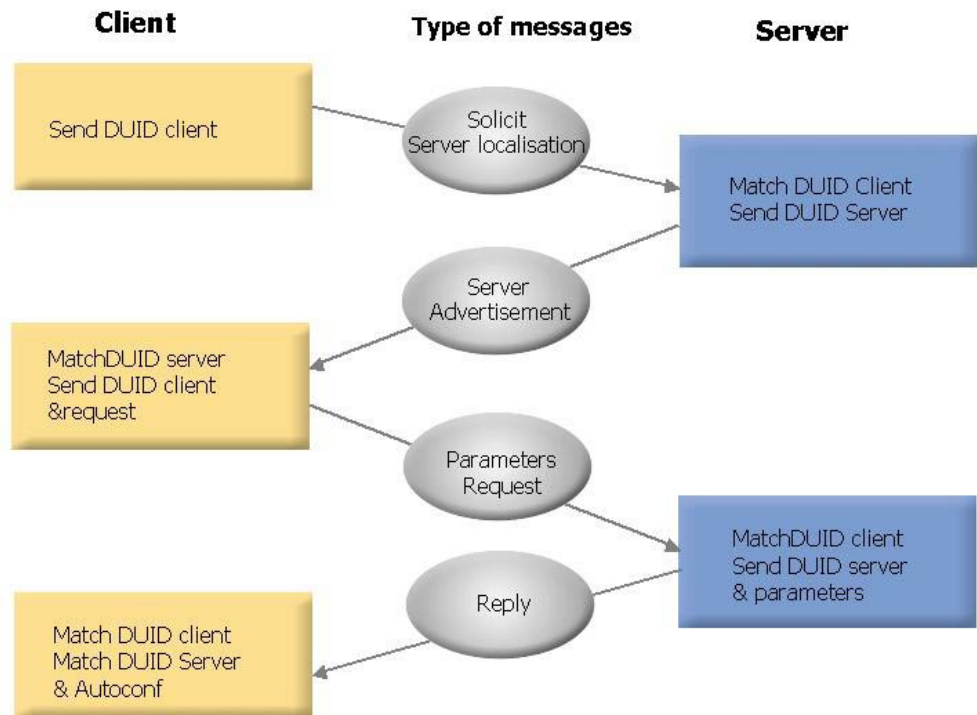


Figure 10. DHCPv6 negotiation

It's the default mechanisms. The statement `rapid_commit` is disabled.

```
6OS{myconfig-dhcpv6}disable rapid_commit
```

If the DHCPv6 server supports it, DHCPv6 exchange can just take 2 messages. The process is described on the Figure 11. DHCPv6 rapid commit.

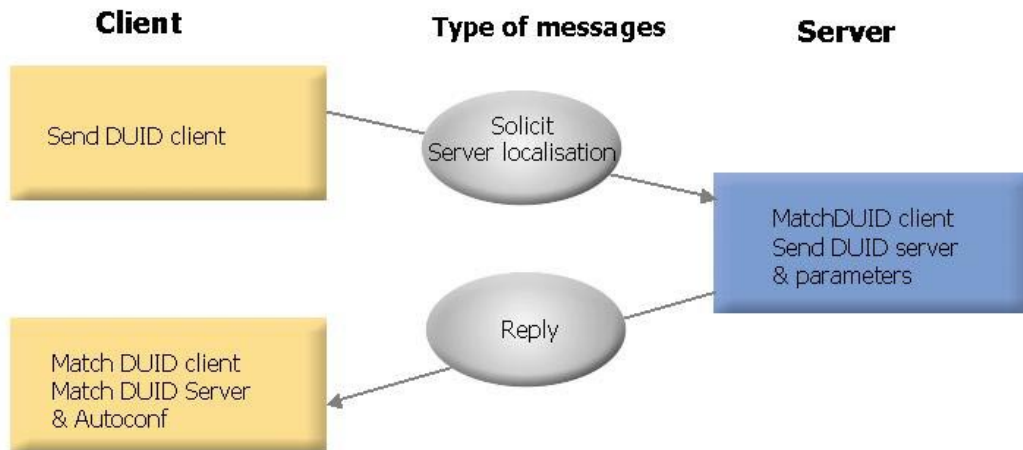


Figure 11. DHCPv6 rapid commit

DNS Server and DNS Proxy Configuration

The DHCPv6 client can be used to ask the address of an IPv6 DNS server.

To ask the DHCPv6 server for IPv6 DNS servers addresses, the following command can be used:

```
6OS{myconfig-dhcpv6}enable dns_request
6OS{myconfig-dhcpv6}disable dns_request
```

The 6WINDGate can act as DNS proxy server. In that case, hosts behind the 6WINDGate can then be configured to send DNS requests to the 6WINDGate DNS proxy. These hosts' resolvers are typically configured to send DNS requests to the DNS well-known site-local addresses. Thus, you may add one of these addresses on the 6WINDGate private interface, with the following command:

```
6OS{myconfig-interf}ipaddress IPv6-site-local-address
```

To enable and disable the 6WINDGate DNS proxy, the following command can be used in the **dns** context:

```
6OS{myconfig-dns}enable proxy
6OS{myconfig-dns}disable proxy
```

Configuration Example

```
6OS{myconfig}dns
6OS{myconfig-dns}enable proxy # DNS proxy function enabled
6OS{myconfig-dns}eth0_0
6OS{myconfig-eth0_0}ipaddress fec0:0:0:fff::1/128 # Add site local address
                                                on the eth0_0 interface

6OS{myconfig- eth0_0}dhcpv6
6OS{myconfig-dhcpv6}enable dhcpv6 # Enable DHCPv6
6OS{myconfig-dhcpv6}interface eth1_0 # The DHCPv6 server can
                                        be reached via eth1_0
6OS{myconfig-dhcpv6}enable dns_request # Enable requests
```

```
6OS{myconfig-dhcpv6}enable prefix_delegation_request # Enable PD function
6OS{myconfig-dhcpv6}sla_id eth0_0 0x0 # Advertises prefix P::/64
6OS{myconfig-dhcpv6}duid 0x000200001ca83033303130323633
```

When DHCPv6 auto-configuration is completed, following actions are performed, given a delegated prefix P::/48:

- A reject route P::/48 is automatically added and marked with the DHCP flag. Any packets whose destination is within P::/48 will never be forwarded outside router's site.

For each interface, a subnet router anycast address is automatically configured P:ID::/64 (ID is specific interface sla-id of dhcpv6 configuration). As 0 is used in the above configuration, 6WINDGate is reachable through IPv6 address Prefix::/64 and will reply with one of its global unicast address.

Chapter 17 Configuring IP Services

This chapter describes how to configure optional IP services.

Configuring IP forwarding

The 6WINDGate makes it possible to forward or not independently the IPv4 and the IPv6 traffic.

Enabling and Disabling IPv4 Forwarding

IPv4 forwarding will be activated by using the following command:

```
60S{myconfig-gen}enable ipv4forwarding
```

IPv4 forwarding will be disabled by using the following command:

```
60S{myconfig-gen}disable ipv4forwarding
```

By default, IPv4 forwarding is enabled.

Enabling and Disabling IPv6 Forwarding

IPv6 forwarding will be activated by using the following command:

```
60S{myconfig-gen}enable ipv6forwarding
```

IPv6 forwarding will be disabled by using the following command:

```
60S{myconfig-gen}disable ipv6forwarding
```


Chapter 18 Configuring IP Services

This chapter describes how to configure optional IP services.

Configuring IP forwarding

The 6WINDGate makes it possible to forward or not independently the IPv4 and the IPv6 traffic.

Enabling and Disabling IPv4 Forwarding

IPv4 forwarding will be activated by using the following command:

```
60S{myconfig-gen}enable ipv4forwarding
```

IPv4 forwarding will be disabled by using the following command:

```
60S{myconfig-gen}disable ipv4forwarding
```

By default, IPv4 forwarding is enabled.

Enabling and Disabling IPv6 Forwarding

IPv6 forwarding will be activated by using the following command:

```
60S{myconfig-gen}enable ipv6forwarding
```

IPv6 forwarding will be disabled by using the following command:

```
60S{myconfig-gen}disable ipv6forwarding
```

Chapter 19 Configuring DNS Proxy and Client

This chapter describes how to configure DNS parameters.

Introduction

DNS (Domain Name Service) provides name to IP address mapping.

DNS-proxy aims at alleviating configuration of hosts on a network when DNS server addresses are for instance dynamically learnt by the equipment (CPE) located at the border of a WAN.

Hosts send all their DNS requests to of the CPE's LAN addresses. The CPE acting as a DNS-proxy will forward these DNS requests to the ISP DNS servers and send back answers to the hosts.

IP version used to transport the host to CPE DNS messages may be different from the version used to transport CPE to DNS-server DNS messages.

A 6WINDGate implements a DNS-proxy.

It also implements a NAT-PT DNS ALG. To have details about this function, please refer to section "Configuring NAT and NAT-PT".

DNS Configuration

Introduction

There are three ways to configure the DNS server addresses to be used by the 6WINDGate DNS-proxy:

- manually (IPv4 and IPv6),
- automatically
 - o in IPv4 thanks to PPP DNS extension,
 - o in IPv6 thanks to the DHCPv6 protocol

Manual Configuration

Configuring DNS server address for the router

The DNS server address can be configured manually using the following command in the **gen** context. The IP address can be an IPv4 or IPv6 one.

```
6OS{myconfig-gen}nameserver ipaddress
```

This type of configuration is used for the routers itself DNS request.

Managing statically DNS request from the hosts

Even if the DNS requests are disabled on the PPP or DHCPv6 context, the 6WINDGate can forward the DNS request from the hosts thanks to the following command in the `dns` context:

```
6OS{myconfig-dns}forwarder DNS_server_address
```

You can delete these entries thanks to the following command:

```
6OS{myconfig -dns}delete forwarder DNS_server_address
6OS{myconfig -dns}delete forwarder all
```

Example:

```
6OS{myconfig-dns}forwarder 1.1.1.1
6OS{myconfig-dns}delete forwarder 1.1.1.1
```

Dynamic Configuration

The 6WINDGate may know the DNS address by two ways, either by PPP in IPv4 or either by DHCP in IPv6.

DNS Proxy configuration

By default, the 6WINDGate behaves like a DNS proxy.

```
6OS{myconfig-dns}enable proxy
```

The DNS proxy can be disabled in using the following command:

```
6OS{myconfig-dns}disable proxy
```

PPP DNS Extensions for IPv4

The IPv4 address of the DNS server can be configured dynamically using the PPP DNS extension (IPCP) in the case of a PPPoE connection using the following command in the `pppoe` context:

```
6OS{myconfig-pppoe0}enable dns_request
```

The DNS request can be disabled in using the following command:

```
6OS{myconfig-pppoe0}disable dns_request
```

For more details about PPP, refer to [Configuring Point To Point Protocols –Page 69](#)

DHCPv6 Protocol for IPv6

The IPv6 address of the DNS server can be configured dynamically via DHCPv6 using the following command in the `dhcpv6` context:

```
6OS{myconfig-dhcpv6}enable dns_request
```

The DNS request can be disabled in using the following command:

```
6OS{myconfig-dhcpv6}disable dns_request
```

For more details about DHCPv6, refer to “Configuring DHCPv6 Client and Prefix Delegation”

Chapter 20 Configuring NAT and NAT-PT

This chapter describes how to configure the NAT and NAT-PT protocols.

Introduction

NAT

Protocol Overview

Although networks can use any IPv4 addresses they want, they could not be connected to the Internet or other remote networks. They must use Internet-legal addresses for applications to function properly and to be routed on the Internet.

However all network administrators have not the luxury of being able to use fully compliant Internet-legal addresses. So Customer Premise Equipment, like 6WINDGates, provide a solution by hiding the IP addresses of the internal devices, making internally generated packets appear as though they are coming from another device that does have an Internet-legal address.

This mechanism is called NAT and is represented with the following figure where the private addresses are translated by the public address to access to the internet.

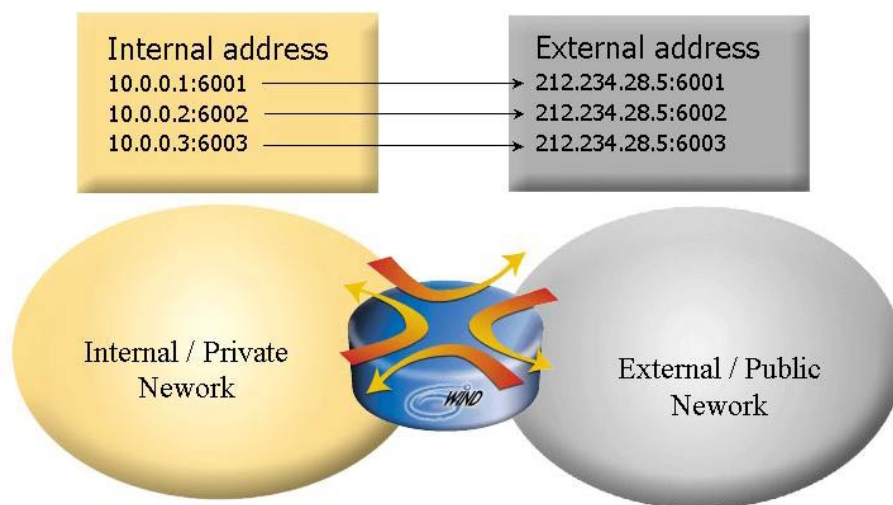


Figure 12. NAT: Address translation

6WINDGate may use a pool of IPv4 public addresses for assignment to translate the private network.

NAT process

The NAT translation changes the source address and port of the outgoing packets and the destination address and port of the incoming packets. In order to perform this translation, the NAT process maintains a translation table.

This table is updated when a new outgoing session is detected. This is the dynamic mode. In this mode, the translation is only possible if the traffic is initiated from the inside part (mono-directional). The lifetime of the translation entries is limited and depends of the state of the session (especially for TCP sessions).

For the dynamic mode, the translation address is always the IPv4 address of the NAT interface (the 6WINDGate's public interface). If necessary, the source port can be changed to avoid confusion between sessions toward the same host and port.

The translation table can also be updated manually by setting static associations. This is the static mode. In this mode, the translation works as well when the traffic is initiated from the inside part as from the outside part (bi-directional). The lifetime of the translation entries is infinite. Static translation rules can be set with address and/or port.

The static and dynamic modes can run simultaneously..

The description of translated flows is set in filtering rules (refer section "Configuring IPv4/v6 Packet Filtering for Firewalls"). The translated packets are diverted to nat (port 8668) then reinjected in the IPv4 stack.

As they are reinjected in the stack, the packets will pass another time in the filter, but skip the rules they have already passed.

Outgoing packets reinjected after being translated skip every filtering rules until rule #65534, which is a "pass all" rule.

Incoming packets reinjected after being translated skip the rules until the first after the one that caused the divert.

As a result, outgoing translated packets pass automatically through filter. Incoming translated packets don't pass automatically through filter.



Caution Encapsulated IPv4 packets (6in4) cannot be sent to NAT.

NAT-PT

Protocol Overview

NAT-PT is the most famous IPv6/IPv4 translation mechanism. It attempts to enable communications between nodes located in an IPv6-only domain and other nodes located in an IPv4-only domain.

The applicability of translation mechanisms is very different from the tunneling mechanisms one. Translation allows communication between IPv6-only hosts and IPv4-only hosts. Tunneling allows communication between IPv6 hosts across IPv4-only domains.

NAT-PT is based on a combination of IP address translation (NAT) and of IPv6/IPv4 protocol translation. It uses a pool of IPv4 addresses for assignment to IPv6 nodes on a dynamic basis as sessions are initiated across IPv4/IPv6 boundaries. The IPv4 addresses are assumed to be globally unique. NAT-PT binds addresses in IPv6 network with addresses in IPv4 network to provide transparent routing for the datagrams traversing between address realms. This requires no changes to end nodes. It does, however, require NAT-PT to track the sessions it supports and mandates that inbound and outbound datagrams pertaining to a session traverse the same NAT-PT router.

NAT-PT also relies on the use of a DNS application level gateway. It uses DNS queries issued by IPv6 hosts in order to set the sessions. Details are provided hereafter.

NAT-PT process

With NAT-PT, outgoing IPv6 packets are translated into IPv4 packets. The source address of the IPv4 packet is the public address configured on the NAT-PT interface. The destination address of the IPv4 packet is deducted from the IPv6 destination address.

As a matter of fact, the packets sent to NAT-PT have a characteristic IPv6 destination address, made of a well-know prefix (the NAT-PT prefix) and the IPv4 address of the destination host. The "DNS-ALG for NAT-PT" paragraph explains how to obtain this characteristic IPv6 destination address.

As for NAT, the NAT-PT process maintains a translation table.

This table is updated when a new outgoing session is detected. This is the dynamic mode. In this mode, the translation is only possible if the traffic is initiated from the inside part (mono-directional). The lifetime of the translation entries is limited and depends of the state of the session (especially for TCP sessions).

The translation table can also be updated manually by setting static associations. This is the static mode. In this mode, the translation works as well when the traffic is initiated from the inside part as from the outside part (bi-directional). The lifetime of the translation entries is infinite. Static translation rules can be set with address and/or port.

The static and dynamic modes can run simultaneously.

The description of translated packets is automatically set in filtering rules (refer to section "Configuring IPv4/v6 Packet Filtering for Firewalls"). The translated packets are diverted to natpt (port 8778) then reinjected in IPv4 or IPv6 stack, depending on the original packet protocol.

As they are reinjected in the stack, the packets will pass another time in the filter, but skip the rules they have already passed.

Outgoing packets reinjected after being translated skip every filtering rules until rule #65534, which is a "pass all" rule.

Incoming packets reinjected after being translated skip rules until they reach the rule after the one that caused the divert.

As a result, outgoing translated packets pass automatically through the filter. Incoming translated packets don't pass automatically through the filter.

DNS-ALG for NAT-PT

When an IPv6 host tries to connect to another host, it will get the destination address through a DNS request. If the destination host is an IPv4 only host, the DNS-ALG intercepts the IPv4 response and translates it into an IPv6 response. So, the IPv6 host receives an IPv6 address as a destination address. The IPv6 address given to the IPv6 host is built from the NAT-PT prefix and the IPv4 destination address.

It is important, for security reasons, to prevent the DNS-ALG from listening to requests that may come from the outside. So the DNS-ALG configuration includes the restricted list of private interfaces to listen to.



Caution Encapsulated IPv4 or IPv6 packets (4in6, 6in6) cannot be sent to NAT-PT.

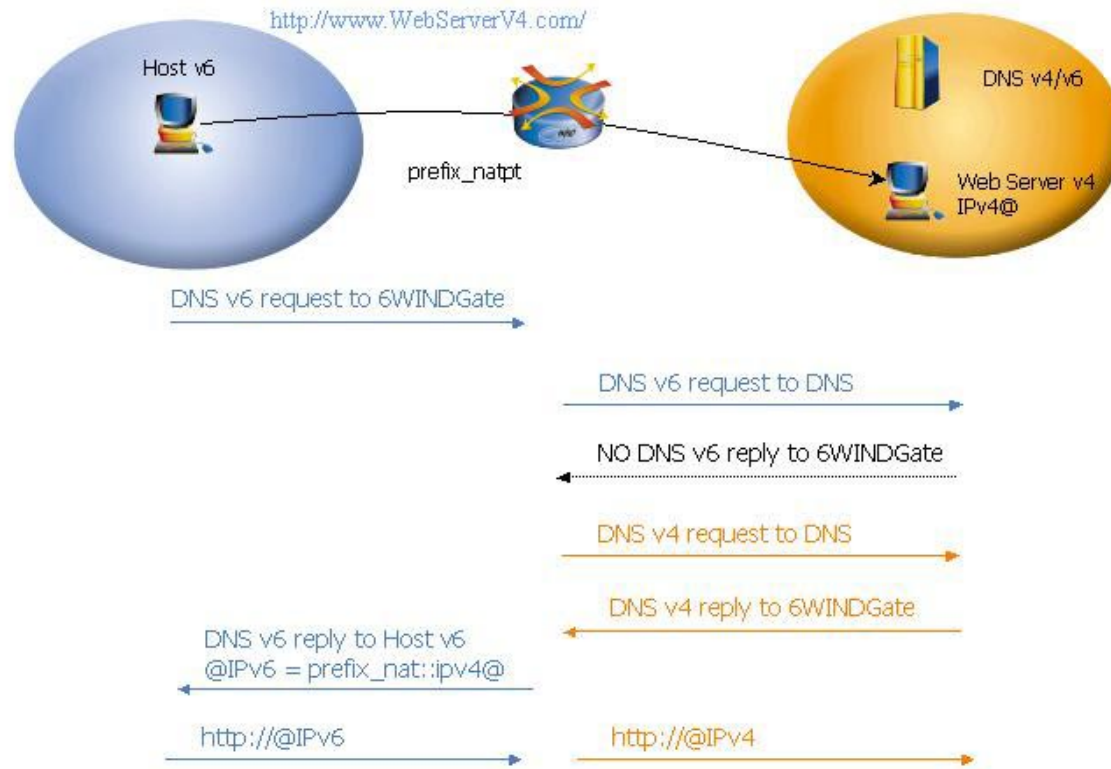


Figure 13. DNS ALG mechanisms with NAT-PT

NAT Context

All the commands related to NAT and NAT-PT configuration have to be entered in the **nat** context.

```
60S{myconfig}nat
```

Displaying NAT and NAT-PT Configuration

To display information about NAT and NAT-PT context, the following commands can be used:

```
60S{myconfig-nat}display
60S{myconfig}display nat
```

Showing NAT and NAT-PT information

To display NAT and NAT-PT information about current status of the 6WINDGate, such as sessions, type:

```
60S{ }show nat
=====
Current Sessions for NAT :

Links at Fri Apr 26 09:17:43 2002

icmp src:10.0.0.10:27322 dst:10.0.103.4:00256 => 212.234.238.114 :27322
      lifetime=60 updated at Fri Apr 26 09:17:39 2002
udp src:224.0.0.9:00520 dst:10.0.103.4:00520 => 224.0.0.9:00520
      lifetime=60 updated at Fri Apr 26 09:17:06 2002
udp src:224.0.0.9:00520 dst:212.234.238.114 :00520 => 224.0.0.9:00520
      lifetime=60 updated at Fri Apr 26 09:17:27 2002
udp src:212.234.238.114 :00520 dst:224.0.0.9:00520 => 212.234.238.114 :00520
      lifetime=60 updated at Fri Apr 26 09:17:27 2002
End of list.

NAT-PT Links at Sat Fri Apr 26 09:17:43 2002

TCP local fec0:1010:5420:cde4:f51f:a20e 3ffe:f00::c365:5e50 1026 80
     remote 195.101.94.80 80.14.9.90 80 49152
     updated at Sat Sep 2 08:45:15 2000
TCP local fec0:1010::5420:cde4:f51f:a20e 3ffe:f00::c365:5e50 1027 80
     remote 195.101.94.80 80.14.9.90 80 49153
     updated at Sat Sep 2 08:45:15 2000
TCP local fec0:1010::5420:cde4:f51f:a20e 3ffe:f00::a05c:7bc1 1068 80
     remote 160.92.123.193 80.14.9.90 80 49195
     updated at Sat Sep 2 08:45:26 2000
TCP local fec0:1010::5420:cde4:f51f:a20e 3ffe:f00::a05c:7bc1 1069 80
     remote 160.92.123.193 80.14.9.90 80 49196
     updated at Sat Sep 2 08:45:26 2000
End of list.
```

Configuring NAT

4 steps are required to configure NAT among which 2 are mandatory to configure NAT in the dynamic mode.

Step 1 Activate NAT.

Using the following command can enable or disable NAT:

```
60S{myconfig-nat}enable nat
60S{myconfig-nat}disable nat
```

By default, NAT is disabled.

Step 2 Define the NAT interface.

To set the NAT interface, use the following command:

```
60S{myconfig-nat}public_interface ifname
```

where:

- *ifname* indicates the interface name or the word **none**

Example

```
60S{myconfig-nat}public_interface eth2_0
```

Note Enabling NAT and defining the NAT interface is enough to configure NAT in dynamic mode, which is the default mode.

Step 3 Define a static association.

Static address translation rules can be defined. Every outgoing packet with Private Address as source address is translated using Public-Address. Every incoming packet with Public-Address as destination address is translated using Private Address.

Use the following command:

```
60S{myconfig-nat}static_association address_v4_public address_v4_private
```

where:

- *address_v4_public* is the public IPv4 address.
- *address_v4_private* is the private IPv4 address.

Example

```
60S{myconfig-nat}static_association 212.234.238.114 10.0.0.1
```

Every received packet on the 6WINDGate with destination address 212.234.238.114 is translated to host 10.0.0.1. This enables incoming or outgoing connections between the private host "10.0.0.1" and any public host.

Step 4 Configure an internal proxy.

A static address translation rule that specifies the port and protocol can also be defined.

Every incoming packet matching the protocol and the port number is translated using Private Address as a destination address. Every outgoing packet matching the protocol and the port number is translated using the NAT interface address as a source address.

To configure an internal proxy, use the following command:

```
60S{myconfig-nat}internal_proxy number port_number protocol address_v4_private
```

where:

- *number* is the number of the rule,
- *port_number* is the port number,
- *protocol* is either *udp* or *tcp*,
- *address_v4_private* is the private address where the packets will be transmitted.

Example

```
60S{myconfig-nat}internal_proxy 1 21 tcp 10.0.0.1
```

Every tcp packet with port 21 (ftp) is translated using the private address 10.0.0.1. This enables any public host to initiate a ftp session with the private host "10.0.0.1" calling the public address "212.234.238.114".

NAT, Filtering and Service Flows

If filtering is disabled, every packet going through the NAT interface is diverted to NAT.

Every other packet (other interfaces) is allowed.

If filtering is enabled, the CLI automatically sets the filter rule that diverts all incoming packets through the NAT interface.

You have to write manually the rules that divert outgoing packet through the NAT interface. Remember that these packets will then pass through the filter.

If service flows are set to “allow” or “clear”, the CLI generates automatically the filter rules to let the corresponding flows pass without being translated. This is done even if the filtering is disabled.

If some service flows should be translated through NAT, they must be set to "translate". The CLI then generates automatically the filter rules to let the corresponding flows being translated and passed.

Configuring NAT-PT

6 steps are required to configure NAT-PT among which 4 are mandatory to configure NAT-PT in the dynamic mode.

Step 1 Activate NAT-PT.

Using the following command can enable or disable NAT-PT:

```
60S{myconfig-nat}enable natpt
60S{myconfig-nat}disable natpt
```

By default, NAT-PT is disabled.

Step 2 Define the NAT-PT interface.

To set the NAT-PT interface in order to select the IPv4 public address used to substitute destination IPv6 addresses, use the following command:

```
60S{myconfig-nat}public_interface ifname
```

where:

- *ifname* indicates the interface name or the word **none**

Example

```
60S{myconfig-nat}public_interface eth2_0
```

Step 3 Set the NAT-PT prefix.

To set the NAT-PT prefix, use the following command:

```
60S{myconfig-nat}natpt_prefix ipv6prefix
```

where:

- *ipv6prefix* is the IPv6 prefix and mask or the word **none**

Example

```
60S{myconfig-nat}natpt_prefix 3ffe:f00::/96
```

Step 4 Define the NAT-PT private interfaces

To set the list of NAT-PT private interfaces on which, the DNS-ALG will listen to DNS requests, use the following commands:

```
60S{myconfig-nat}natpt_domain_interfaces ifname1 ifname2
```

where:

- *ifnameX* indicates an interface name.

```
60S{myconfig-nat}natpt_domain_interfaces automatic | none
```

If the keyword **automatic** is used instead of the interface list, the DNS-ALG will listen to all interfaces except the public interface.

If the keyword **none** is used instead of the interface list, the DNS-ALG is not activated.

Example

```
60S{myconfig-nat} natpt_domain_interfaces eth0_0 eth1_0
```



Note

The minimum configuration for NAT-PT includes: enabling NAT-PT, defining the NAT-PT interface, setting the NAT-PT prefix and defining the NAT-PT private interfaces. This will run in dynamic mode, which is the default mode.

Step 5 Define a static association

Static address translation rules can be defined. Every outgoing packet with Private Address as source address is translated using Public Address. Every incoming packet with Public Address as destination address is translated using Private Address.

Use the following command:

```
60S{myconfig-nat}static_association address_v4_public address_v6_private
```

where:

- *address_v4_public* is the public IPv4 address.
- *address_v6_private* is the private IPv6 address.

Example

```
60S{myconfig-nat}static_association 212.234.238.114 3ffe:eff3::58
```

Every received packet on the 6WINDGate with destination address 212.234.238.114 is translated to host 3ffe:eff3::58. This enables incoming or outgoing connections between the private host "3ffe:eff3::58" and any public host.

Step 6 Configure an internal proxy.

A static address translation rule that specifies the port and protocol can also be defined.

Every incoming packet matching the protocol and the port number is translated using PrivateAddress as a destination address.

Use the following command:

```
60S{myconfig-nat}internal_proxy number port_Number protocol address_v6_private
```

where:

- *number* is the rule number,
- *port_Number* gives the packet protocol,
- *protocol* defines the protocol, it can be *tcp*, *udp* or *any*. The default value is *any*,
- *address_v6_private* is the private IPv6 address.

Example

```
60S{myconfig-nat}internal_proxy 1 21 tcp 3ffe:eff3::64
```

Every tcp packet with port 21 (ftp) is translated using the private address 3ffe:eff3::64. This enables any public host to initiate an ftp session with the private host "3ffe:eff3::64" via the public address "212.234.238.114".

NAT-PT and Filtering

Every IPv6 packet going through the NAT-PT interface with a destination address matching the NAT-PT prefix and mask, is diverted to the NAT-PT.

Every other packet (other interfaces) is allowed.

If filtering is enabled, the CLI automatically sets the filter rule that diverts incoming and outgoing packets through the NAT-PT interface.

If security tunnels are defined (see section Configuring IPv4/v6 IPsec– Page 138), the CLI generates automatically the filter rules to let the corresponding flows go through without being translated.

Routing the NAT-PT Prefix

When NAT-PT is enabled, an IPv6 mapped address is automatically set on the NAT-PT public interface. This address is built from the NAT-PT prefix and the loopback address.

As a result, if a routing protocol is set on the private interfaces, the route to the NAT-PT interface will be advertised on the private network.

Configuration Examples

NAT Configuration Example

Let's consider the case described in **Erreur ! Source du renvoi introuvable.** where a 6WINDGate interconnects two IPv4 sites with an IPv4 WAN. On the first LAN, there are two machines (Host 1 and Host 2). A FTP server (Host 3) is placed on the second LAN.

Below, we are going to describe the connections between these three machines and a host on the WAN:

- Connection 1: the session is open from Host 1 to Host 4. It is the typical case with dynamic NAT.
- Connection 2: the session is open from Host 4 to Host 2. The destination address of packets from Host 4 is 212.234.238.114. This one is an address of the 6WINDGate address pool. The 6WINDGate is configured to redirect the traffic to Host 2. In this case, a static association is defined.
- Connection 3: Host 4 opens an FTP session on Host 3. A static address translation rule specified with the port and protocol is defined on the 6WINDGate.

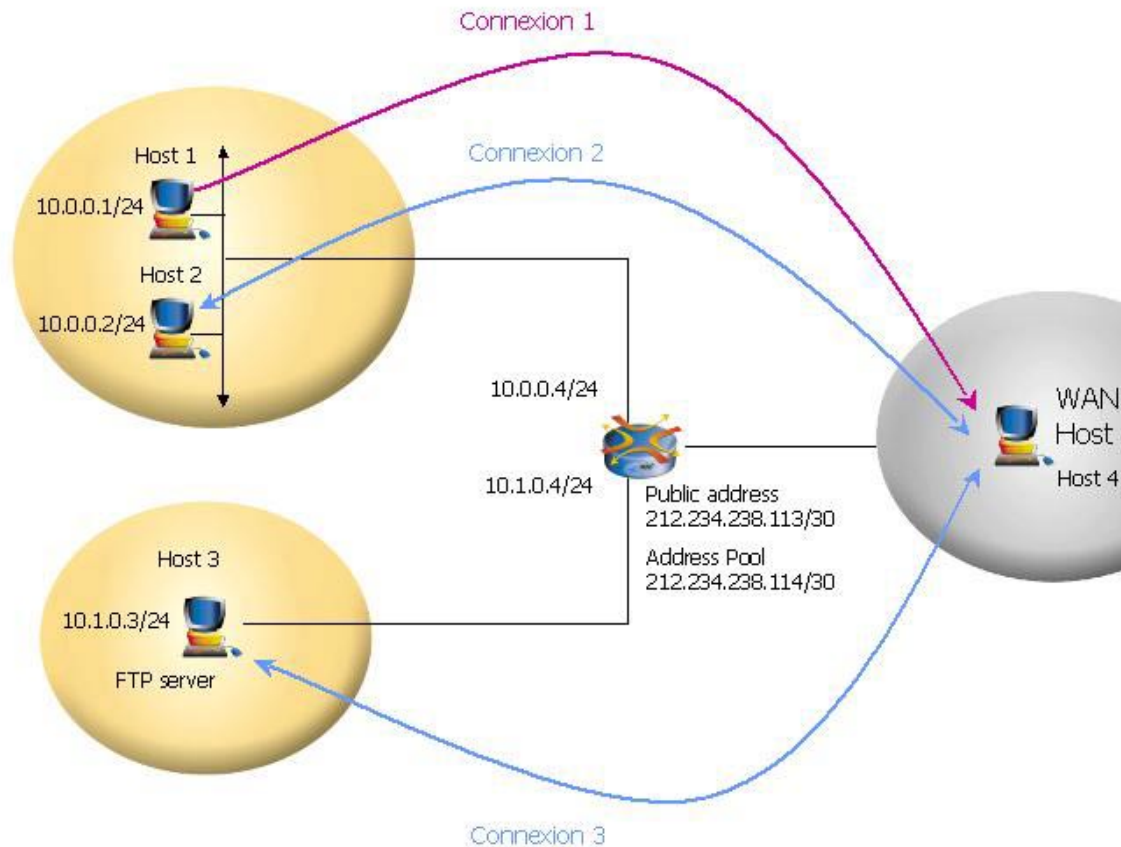


Figure 14 Three connection examples with NAT

The configuration of the 6WINDGate corresponding to this example will be:

- For the private interface eth0_0:

```
60S{myconfig-eth0_0}ipaddress 10.0.0.4/24
```
- For the optional interface eth1_0:

```
60S{myconfig-eth1_0}ipaddress 10.1.0.4/24
```
- For the public interface eth2_0:

```
60S{myconfig-eth2_0}ipaddress 212.234.238.113/30
```
- Enter in the NAT context:

```
60S{myconfig-eth2_0}nat
```

- Enable NAT and define the public NAT interface. Host 1 will be seen on the internet with the address 212.234.238.113.

```
60S{myconfig-nat}enable nat
60S{myconfig-nat}public_interface eth2_0
```



Note Defining the NAT interface and enabling NAT is enough to configure NAT in dynamic mode, which is the default mode. It matches the connection example 1.

- Create a static association for the connection 2. Host 4 may open a session to Host 2 if the packet destination address is 212.234.238.114. The packets will be automatically forwarded to Host 2 (10.0.0.2)

```
60S{myconfig-nat}static_association 212.234.238.114 10.0.0.2
```

- Create a static address translation rule on port 21 and tcp protocol. The packets will be automatically forwarded to Host 3 (10.1.0.3)

```
60S{myconfig-nat}internal_proxy 1 21 tcp 10.1.0.3
```

NAT-PT Configuration Example

Let's consider the case described in Figure 13 where a 6WINDGate interconnects two private IPv6 sites with an IPv4 WAN. On the first LAN, there are two machines (Host1 and Host 2). An FTP server (Host 3) is placed on the second LAN.

Below, we are going to describe the connections between these three machines and a host on the WAN:

- Connection 1: the session is open from Host 1 to Host 4. It is the current case of dynamic NAT-PT.
- Connection 2: the session is open from Host 4 to Host 2. The destination address of packets from Host 4 is 212.234.238.114. This one is an address of the 6WINDGate addresses pool. The 6WINDGate is configured to redirect the traffic to the Host2 in this case. A static association is defined.
- Connection 3: Host 4 opens an FTP session to Host3. A static address translation rule specifying the port and protocol is defined on the 6WINDGate.

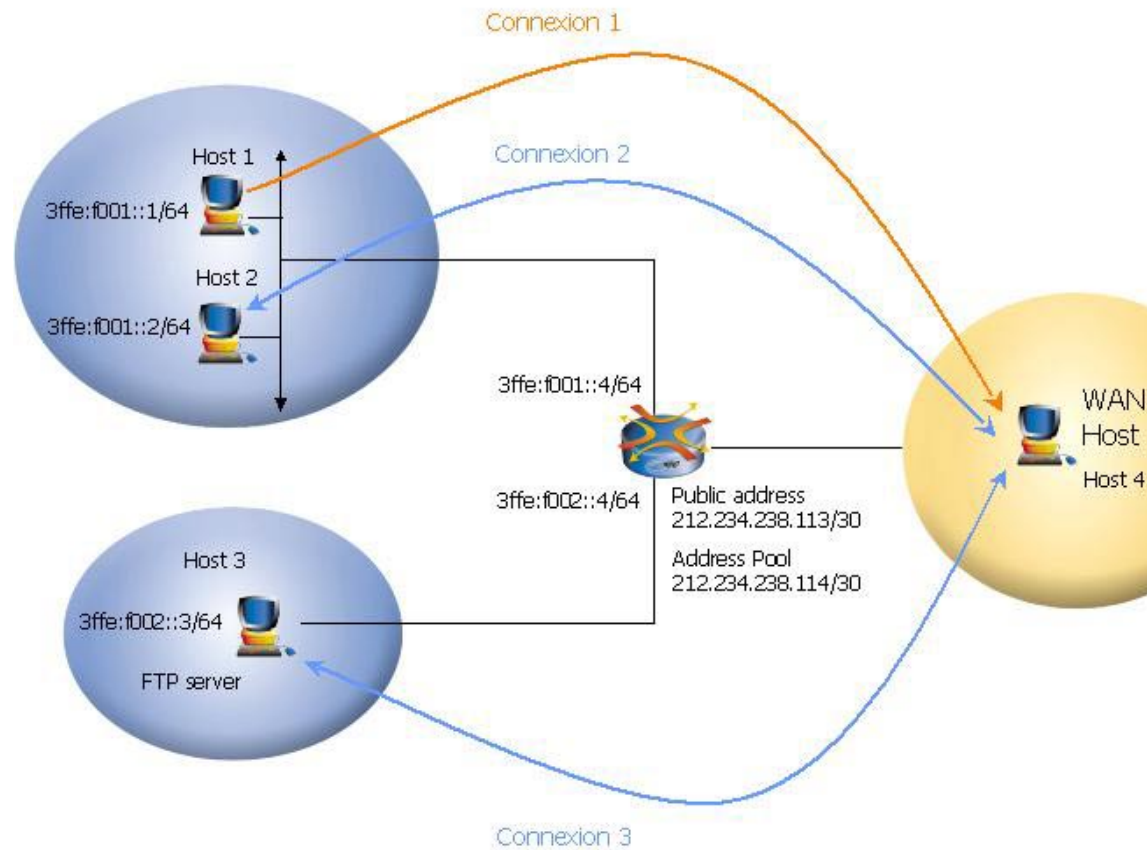


Figure 15 Three connections examples with NAT-PT

The configuration of the 6WINDGate corresponding to this example will be:

- For the private interface eth0_0
`60S{myconfig-eth0_0}ipaddress 3ffe:f001::4/64`
- For the optional interface eth1_0
`60S{myconfig-eth1_0}ipaddress 3ffe:f002::4/64`
- For the public interface eth2_0
`60S{myconfig-eth2_0}ipaddress 212.234.238.113/30`
- Enter in the NAT context
`60S{myconfig-eth2_0}nat`
- Enable NAT and define the public NAT interface. Host 1 will be seen on the internet with the address 212.234.238.113.
`60S{myconfig-nat}enable natpt`
`60S{myconfig-nat}public_interface eth2_0`
`60S{myconfig-nat}natpt_prefix 2200:3456:0:0:7788:99AA::0/96`
`60S{myconfig-nat}natpt_domain_interfaces eth0_0 eth1_0`



Note The minimum configuration for NAT-PT includes: enabling natpt, defining the NAT-PT interface, setting the NAT-PT prefix and defining the NAT-PT private interfaces. This will run in dynamic mode, which is the default mode.

- Connection 2: create a static association. Host 4 may open a session to Host 2 if the packets destination address is 212.234.238.114. The packets will be automatically forwarded to Host 2 (3ffe:f001::2)

```
60S{myconfig-nat}static_association 212.234.238.114 3ffe:f001::2
```

- Create a static address translation rule on port 21 and tcp protocol. The packets will be automatically forwarded to Host 3 (3ffe:f002::3)

```
60S{myconfig-nat}internal_proxy 1 21 tcp 3ffe:f002::3
```


Chapter 21 Configuring IPv6 Mobility

This chapter describes how to configure IPv6 mobility functions.

Introduction

Figure 16 describes the different equipment involved in Mobile IPv6 and summarises the different steps occurring when a node moves from a location to another.

The 6WINDGate implementation complies with the IETF draft 20 of Mobile IPv6.

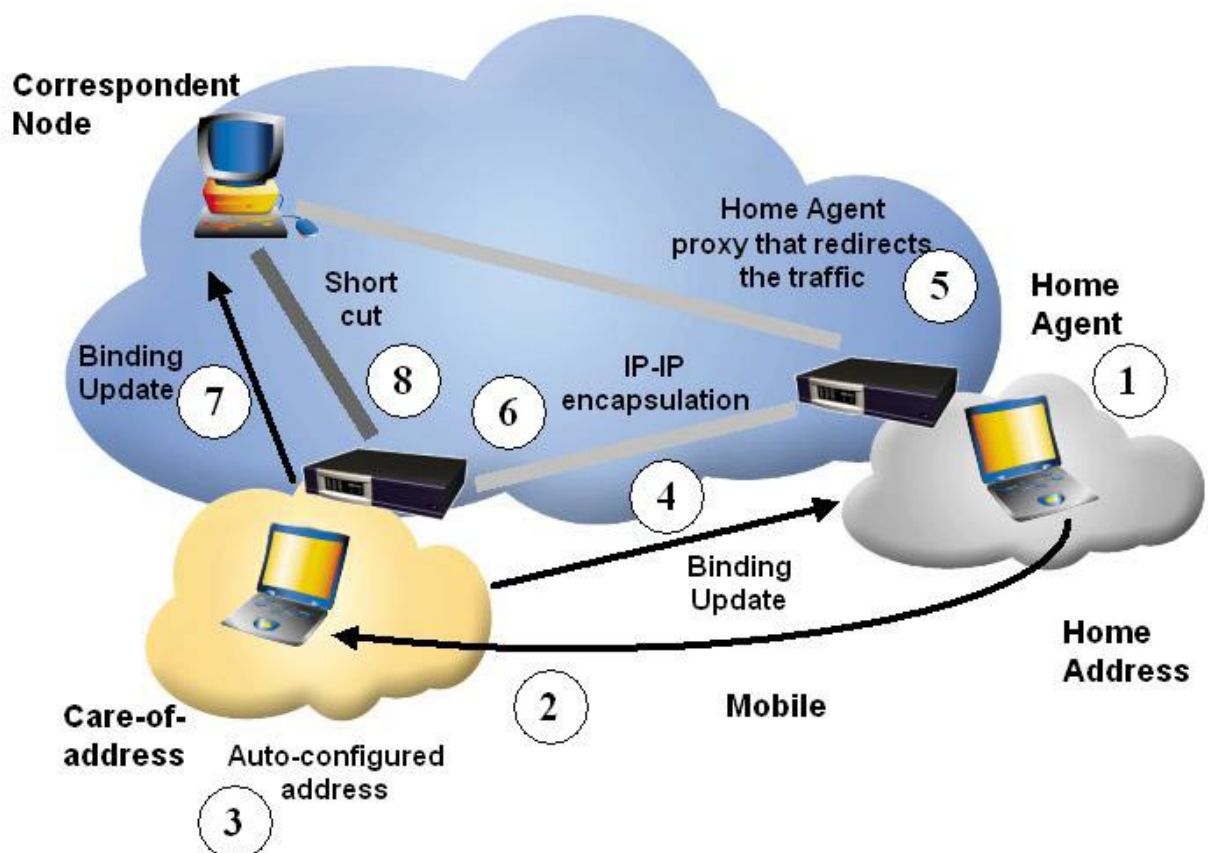


Figure 16: Mobile IPv6 architecture

These different steps are as follow:

- The mobile node is located in its Home Network and declared to its Home Agent. Its address is a permanent one called Home Address. A communication is taking place between the mobile node and its Correspondent.
- The mobile node is moving from its location to another.
- Once located in a Foreign Network, the IPv6 auto-configuration process between the local router and the mobile node takes place and the mobile gets a temporary address called Care-of-Address.
- The mobile notifies its Home Agent of its new address; this step is called a Binding Update. The Home Agent updates a local Binding Cache with the information.
- Now the Home Agent is able to redirect the traffic coming from the Correspondent that is not aware of the new location of the mobile node.
- To do so, the Home Agent uses an IP in IP encapsulation technique.
- When receiving an encapsulated traffic, the mobile node notifies the Correspondent of its new address with a second Binding Update. The Correspondent updates its local Binding Cache with the information.
- A direct communication (short-cut) is now possible between the Correspondent and the mobile node.

In the Mobile IPv6 architecture, the 6WINDGate acts as a Home Agent or as a Correspondent Node.

Configuring IPv6 Mobility



Note Please note that Mobile IPv6 is still in the standardisation process at the IETF. The present SixOS implementation complies with the draft 20 of Mobile IPv6.

MIP Context

IPv6 mobility parameters can be configured in the `mip` context.

Configuring IPv6 Autoconfiguration Mechanisms

As IPv6 mobility relies on the IPv6 auto-configuration mechanism, it has to be configured first. To do so, please refer to section [IPv6 Auto Configuration for a router](#) – Page 48.

In order to help the mobile movement detection, it's hardly recommended to send Router Advertisements every 1 second on each network the mobile may move. In the same way, the advertisement interval option is recommended to be diffused in the Router Advertisement.

```
ED1{myconfig-eth0_0}router_advert smart 1000 1800 address 1460 yes
```

Displaying Mobile IP Information

Mobile IPv6 status can be displayed using the following commands:

```
ED1{} show service
ED1{} show interface
```

Examples

```
ED1{} show service
Service SSH is active
Service SNMP is inactive
Service NAT is inactive
Service NATPT is inactive
Service FILTER is inactive
Service RIP is inactive
Service RIPng is inactive
Service OSPFv2 is inactive
Service OSPFv3 is inactive
Service BGP is inactive
Service TELNET is active
Service IP Forwarding is active
Service IPv6 Forwarding is active
Service IPSEC is inactive
Service DHCPSEVER is inactive
Service Mobility node type: home agent
...
ED1{}
```

The **show service** command shows whether or not the 6WINDGate is acting as a Home Agent or not.

```
ED1{} show interface
eth0_0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.64.1.201 netmask 0xffffffff broadcast 10.64.1.255
    inet6 fe80::290:fbff:fe04:71b%eth0_0 /64
    inet6 3ffe:304:124:6401::201 /64
    inet6 3ffe:304:124:6401:290:fbff:fe04:71b /64
    ether 00:90:fb:04:07:1b
    media: Ethernet autoselect (10baseT/UTP <full-duplex>)
eth2_0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.63.1.201 netmask 0xffffffff broadcast 10.63.1.255
    inet6 fe80::5054:40ff:fe20:6df0%eth2_0 /64
    inet6 3ffe:304:124:6301::201 /64
    inet6 3ffe:304:124:6301:5054:40ff:fe20:6df0 /64
    ether 52:54:40:20:6d:f0
    mip6 Home Agent
eth1_0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.16.0.233 netmask 0xffffffff broadcast 10.16.0.255
    ether 00:50:04:36:8c:79

Automatic tunnel:
tu0: flags=1041<UP,RUNNING,LINK0> mtu 1480
    inet6 ::127.0.0.1 /96
```

```
lladdr 7f:00:00:01
ED1{}
```

The **show interface** command is more specific and shows on which interface the 6WINDGate is acting as a Home Agent.


Modifying the Mobile IP behaviour

The 6WINDGate is configured as a home agent with the following command.

```
ED1{myconfig-mip}home_agent interface
```

where:

- *interface* is a valid interface on which, at less one IPv6 prefix is advertised. This IPv6 prefix is the Home Prefix and the interface refers to the Home Network.



Note If no prefix is advertised, the SixOS don't act as an Home Agent.

The 6WINDGate can act as Home Agent on more than one interface. To add another interface, the last command has to be entered again with the other interface this time.

Example:

```
ED1{myconfig-mip}display
# MOBILITY HOME AGENTS

ED1{myconfig-mip}home_agent eth2_0
ED1{myconfig-mip}home_agent eth1_0
ED1{myconfig-mip}display
# MOBILITY HOME AGENTS
home_agent eth2_0
home_agent eth1_0

ED1{myconfig-mip}
```

To disable the Home Agent mobility function, the following command has to be entered for each **home_agent** command previously configured:

```
ED1{myconfig-mip}delete home_agent interface
```

Example:

```
ED1{myconfig-mip}display
# MOBILITY HOME AGENTS
home_agent eth2_0
home_agent eth1_0

ED1{myconfig-mip}delete home_agent eth2_0
ED1{myconfig-mip}delete home_agent eth1_0
ED1{myconfig-mip}display
# MOBILITY HOME AGENTS

ED1{myconfig-mip}
```

Managing the Mobile Node Binding Cache Entries

To display the present mobile registered on the Home Agent, the following command has to be entered:

```
ED1{ }show mobiles

Mobile 1:
  Home Address:          3ffe:304:124:6200:250:fcff:fe6e:506d
  Care-of Address:      3ffe:304:124:6400:250:fcff:fe70:6fe0
  Home Agent Address:    3ffe:304:124:6200::104
  Flags  Sequence Number Lifetime  Lifetime Expired  State
  AHL-          57558      420          420          ---
Mobile 2:
  Home Address:          3ffe:304:124:2201::201
  Care-of Address:      3ffe:304:124:6401:a2b0:56ff:fe5c:12c3
  Home Agent Address:    3ffe:304:124:2201::102
  Flags  Sequence Number Lifetime  Lifetime Expired  State
  ----          36      420          105          ---
```

where :

- **flags** is the list of flags used in the received Binding Update

A : Acknowledge

H : Home registration

L : Link-Local Address Compatibility

K : Key Management Mobility Capability

To delete an entry (or all) in the binding cache list, enter the following command.

```
ED1{ }flush mobile {IPv6address|all}
```

where:

- *IPv6address* is the peer Home Address (Phaddr) associated with the binding cache entry.
- **all** flushes all peer Home Addresses.

Example:

```
ED1{ }flush mobile 3ffe:304:124:2201::102
ED1{ }flush mobile all
```

Chapter 22 Configuring SNMP

This chapter describes how to configure SNMP and how to use the different MIBs supported by the 6WINDGate.

Configuring SNMP Support

Overview

The Simple Network Management Protocol (SNMP) system consists of the following three parts:

- A SNMP manager
- A SNMP agent
- A Management Information Base (MIB)

SNMP is an application-layer protocol that provides a message format for communication between SNMP managers (clients) and agents (servers).

The SNMP manager can be part of a Network Management System. The agent and MIB reside on the 6WINDGate to monitor. To configure SNMP on the 6WINDGate, you have to define the relationship between the manager and the agent.

The 6WINDGate implements the version 2 of the SNMP protocol.

Enabling and Disabling SNMP

The user can enable or disable the SNMP agent by using respectively the two following commands in the `snmp` context:

```
6OS{myconfig-snmp}enable snmp
6OS{myconfig-snmp}disable snmp
```

By default, the SNMP agent is disabled.

SNMP Context

All commands for configuring the relationship between the manager and the agent have to be typed in the `snmp` context.

Creating and Modifying Access Control for an SNMP Community

Use an SNMP community string to define the relationship between the SNMP manager and the agent. The community string acts like a password to permit access to the 6WINDGate MIBs.

The following commands create read-only and read-write communities:

```
6OS{myconfig-snmp}rocommunity community [source]
6OS{myconfig-snmp}rwcommunity community [source]
```

where:

- *community* is a string which acts like a password to permit access on the 6WINDGate
- *source* can be a hostname, a subnet, or the word *default*. A subnet can be specified as IP/MASK or IP/BITS.

Establishing the sysContact and sysLocation

You can set the system contact and location of the SNMP agent. This information is reported by the 'system' table in the mib-II tree. To do so, use the following commands:

```
6OS{myconfig-snmp}syslocation string
6OS{myconfig-snmp}syscontact string
```

Configuring SNMP Traps

The following command defines the default community string to be used when sending traps.

```
6OS{myconfig-snmp}trapcommunity community
```

The following commands define the hosts to which traps (and/or inform notifications) should be sent. The 6WINDGate sends a Cold Start trap when the SNMP agent starts up. If enabled, it also sends traps on authentication failures. Multiple **trapsink**, **trap2sink** and **informsink** lines may be specified to specify multiple destinations. Use **trap2sink** to send SNMPv2 traps and **informsink** to send inform notifications. If *community* is not specified, the string from a preceding **trapcommunity** statement will be used. If *portnumber* is not specified, the well-known SNMP trap port (162) will be used.

```
6OS{myconfig-snmp}trapsink host [community] [port portnumber]
6OS{myconfig-snmp}trap2sink host [community] [port portnumber]
6OS{myconfig-snmp}informsink host [community] [port portnumber]
```

Setting **authtrap** to *enable* enables generation of authentication failure traps. Setting **authtrap** to *disable* disables them.

The user can display the configuration of SNMP parameters by using the following commands:

```
6OS{myconfig}display snmp
6OS{myconfig-snmp}display
```

SNMP Configuration Example

The following example grants read-only access to the whole MIB to the host 10.0.0.117 using the community *6wind*. No other SNMP manager will have access to the agent.

SNMP Authentication Failure and all other traps are sent via SNMPv2 to the host 10.0.0.117 using the community string *public*. The **syslocation** and **syscontact** variables are defined.

```
6OS{myconfig-snmp}rocommunity 6wind 10.0.0.117
6OS{myconfig-snmp}syslocation "Montigny le Bretonneux"
6OS{myconfig-snmp}syscontact network@6wind.com
```

```
6OS{myconfig-snmp}trapcommunity public
6OS{myconfig-snmp}authtrap enable
6OS{myconfig-snmp}trap2sink 10.0.0.117
```

Supported MIBs

This paragraph lists MIBs supported by the 6WINDGate

The 6WINDGate conforms to:

- RFC 1231: MIB II
- RFC 2452: IPv6 Management Information Base for TCP
- RFC 2454: IPv6 Management Information Base for UDP
- RFC 2465: Management Information Base for IPv6: Textual Conventions and General Group
- RFC 2466: Management Information Base for IPv6: ICMPv6 Group
- draft-ietf-ipv6-rfc2011-update-02: Management Information Base for the Internet Protocol
- RFC 1907: Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)

Chapter 23 Configuring IPv4/v6 Quality of Service

This chapter describes how to configure IPv4 and IPv6 QoS mechanisms.

Introduction

Configuring the 6WINDGate QoS mechanisms requires some knowledge of the basic principles of IP Differentiated Services (DiffServ).

Prior reading of the *6WIND QoS management – White paper* may be helpful.

Definitions

Class

In the IETF “Differentiated Services” (DiffServ) approach, as no signaling protocol is used, each packet is tagged with a “traffic class” that makes the implementation of traffic differentiation possible in the network. The way to manage a class in each node is defined by a PHB.

Flow

A flow is defined by a combination of one or more IP header fields, such as source address, destination address, protocol ID, source port and destination port numbers, and other information such as incoming interface. Once defined, the class a flow belongs to is determined.

PHB

A PHB (Per Hop Behavior) is a description of the externally observable forwarding behavior of a DiffServ node applied to particular flows in a class.

PHBs are implemented in nodes by means of some buffer management and packet scheduling mechanisms. PHBs are defined in terms of behavior characteristics relevant to service provisioning policies, and not in terms of particular implementation mechanisms.

At the present time, the IETF has standardized two PHBs: Expedited Forwarding and Assured Forwarding.

Expedited Forwarding

The EF (Expedited Forwarding) PHB can be used to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through DiffServ domains. Such a service appears to the endpoints like a point-to-point connection or a “virtual leased line”.

Assured Forwarding

There is a demand to provide assured forwarding of IP packets over the Internet. In a typical application, a company uses the Internet to interconnect its geographically distributed sites and wants an assurance that IP packets within this intranet are forwarded with high probability as long as the aggregate traffic from each site does not exceed the subscribed information rate (profile). It is desirable that a site may exceed the subscribed profile with the understanding that the excess traffic is not delivered with as high probability as the traffic that is within the profile. It is also important that the network does not reorder packets that belong to the same microflow, no matter if they are in or out of the profile.

An AF (Assured Forwarding) PHB group is a means to offer different levels of forwarding assurances for IP packets received from a customer.

Four AF classes are defined. Within each AF class, IP packets are marked with one of three possible drop precedence values. In case of congestion, the drop precedence of a packet determines the relative importance of the packet within the AF class.

Class Template

Class templates describe some predefined classes of service. The class templates correspond to the five PHBs defined by the IETF for Differentiated Services, namely EF, AF1, AF2, AF3 and AF4. The template alleviates the work of programming the QoS classes.

QoS Context

QoS may be enabled on a per interface and per direction basis, in a logical interface context. The rest of QoS configuration is performed in the `qos` context. For forward compatibility reasons, some parameters may still be configured in a logical interface context, for instance setting the maximum IP bandwidth on the interface

Displaying QoS Configuration

To display information about QoS functions, the following commands can be used:

```
6OS {myconfig-qos} display
```

```
6OS {myconfig} display qos
```

QoS Configuration Steps

The configuration of QoS on a 6WINDGate requires several actions:

- Step 1 Identification of QoS requirements. The user will have to identify the available bandwidth on interface, what kind of services will have to be provided, which flows will be processed by the QoS management.
- Step 2 Enabling QoS management is described in section Enabling QoS Management – Page 123.
- Step 3 Definition of the maximum available bandwidth per interface described in section Defining a Maximum Bandwidth – Page 124.
- Step 4 Defining what kind of services will have to be provided leads to determine which PHB and which class can be used. The user can choose between the two IETF PHBs (EF and AF) and the five associated classes (EF, AF1, AF2, AF3 and AF4) implemented in the 6WINDGate. A class has to be defined for ingoing and outgoing traffic. For each of the five classes, 6WIND provides a class template. The best effort class is always available. This step is described in section Defining a QoS Class – Page 124.
- Step 5 Definition of the flows that will be processed by the QoS management. Flows are defined thanks to filters on parameters such as source and destination addresses, source and destination ports, and protocol type. The flows are bound to classes. At this step, the user defines which class a flow belongs to. It also defines the flow drop precedence and the way its DS byte will be marked. This step is described in section Defining a QoS Flow – Page 125.
- Step 6 Apply the configuration to make it the running configuration.
- Step 7 If the current configuration behaves as expected, make it active at next boot time.

Enabling QoS Management

IP QoS management can be enabled or disabled per interface in a specified way. By default, QoS management is disabled. Enabling and disabling QoS management has to be done in the interface context.

Enabling QoS management on an interface is done with the following command:

```
6OS{myconfig-ifname}enable qos way
```

where :

way is set to the direction of the direction (in or out).

Disabling QoS management on an interface is done with the following command:

```
6OS{myconfig-ifname}disable qos way
```

Example:

```
6OS{myconfig-eth1_0}enable qos in
6OS{myconfig-atm3_0_pvc2}disable qos out
```

Defining a Maximum Bandwidth

A maximum allowed bandwidth per interface can be defined using the following command in the **qos** context:

```
6OS {myconfig-qos} maxbandwidth interface way value unit
```

where:

interface should be set to the logical interface name.

way should be set to the direction of the flow (*in* or *out*).

value is set to the maximum allowed bandwidth for the interface in the specified direction.

unit expresses the bandwidth in unit: *bps*, *Kbps*, *Mbps*. This expression is case sensitive.

or by using the “compact form” with others parameters:

```
6OS {myconfig-qos} intf interface way bw="bandwidth unit"  
                  be_queue="queue_size unit" [{keep_dscp|remark=value}]
```

Defining a QoS Class

QoS classes can be defined using the following command:

```
6OS {myconfig-qos} class name template_name interface way bandwidth unit [queue_size unit]  
[keep_dscp]
```

where:

name is the name given to the class

template_name should be set to the name of the template to be used by the class. The template should be selected in the following list:

- EF**: Expedited Forwarding.
- AF1**: Assured Forwarding 1.
- AF2**: Assured Forwarding 2.
- AF3**: Assured Forwarding 3.
- AF4**: Assured Forwarding 4.

Refer to the section Class Template Presentation –Page 126 to know more about QoS parameters selected for **6WIND** QoS templates.

interface should be set to the logical interface name.

way should be set to direction of the flow (*in* or *out*).

bandwidth is set to the allowed bandwidth for the class.

unit expresses the bandwidth in *bps*, *Kbps*, *Mbps*. This expression is case sensitive.

queue_size is the value of the buffer size allowed to the flow. This parameter is optional. If not mentioned, the burst parameter is automatically calculated in order to buffer size 500 ms of traffic for an EF class and 5 s to an AF one. For configuring this parameter, please refer to section Configuring a Class Buffer Size - Page 128;

unit expresses the buffer size in *B*, *KB*, *MB*. This expression is case sensitive.

keep_dscp indicates that the DSCP value should remain unchanged.

The best effort class is always available.

An existing class can be deleted using the following command:

```
6OS {myconfig-qos} delete class classname
```

All defined classes can be deleted in a single operation the following command:

```
6OS{myconfig-qos} delete class all
```

Example:

```
6OS{myconfig-qos} class EF_out EF eth0_0 out 200 Kbps
6OS{myconfig-qos} class AF1_out AF1 eth1_0 out 200 Kbps 50 KB
6OS{myconfig-qos} class AF2_out AF2 eth0_0 out 200 Kbps
6OS{myconfig-qos} class AF3_out AF3 eth0_0 out 200 Kbps keep_dscp
6OS{myconfig-qos} delete class AF1_out
```

Defining a QoS Flow

Flows can be defined using the following command:

```
6OS{myconfig-qos} flow flowname source destination protocol [dscp=value] classname[/drop
precedence] [{keep_dscp|remark=value}]
```

where:

flowname is the name given to the flow.

source and *destination* filter the source and destination of the traffic. IPv4 or IPv6 addresses are accepted. The format is one of:

```
address/mask[port]
address/mask
address[port]
address
```

port can be a number or a string.

By default, *mask* parameter is set to 32 for IPv4 or 128 for IPv6. When a flow applies to any traffic, the *0.0.0.0/0* (respectively *::/0*) notation can be used for IPv4 (respectively IPv6).

protocol indicates the type of protocol selected. It can be *udp*, *tcp* or *any* (default value).

dscp=value indicates the DSCP values that the packets of the flow match. It can be a binary value (%101), a hexadecimal value (0x2B), a decimal value (12) or a predefined value (EF, AF11, AF12, AF13, AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43).

classname is the name of the class in which the packets of the flow will be scheduled.

drop_precedence is optional and is only used for AF PHBs to set the drop precedence of the packets. This level can be **low**, **medium** or **high** and corresponds to the three levels defined by the IETF.

keep_dscp indicates that the DSCP marker should remain unchanged for this flow

remark=value means the DSCP will be remarked for this flow. It can be a binary value (%101), a hexadecimal value (0x2B), a decimal value (12) or a predefined value (EF, AF11, AF12, AF13, AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43).

An existing flow can be deleted using the following command:

```
6OS{myconfig-qos} delete flow flowname
```

All defined flows can also be deleted using the following command:

```
6OS{myconfig-qos} delete flow all
```

Example:

```
6OS {myconfig-qos} flow WEB_IN 3ABC:1::/64[http] 3DEF:1::/64 tcp AF1_in/high
6OS {myconfig-qos} flow MAIL_IN 3ABC:1::/64[smtp] 3DEF:1::/64 tcp AF1_in/low
6OS {myconfig-qos} flow UNCHG 3ABC:2::/64 3DEF:2::/64 tcp dscp=EF EF keep_dscp
6OS {myconfig-qos} flow CHG 123::/64 4321::/64 tcp dscp=AF11 class_name remark=AF12
6OS {myconfig-qos} flow Class_Change ::/0 ::/0 any dscp=EF class_name remark=AF11
6OS {myconfig-qos} delete flow WEB_IN
```



Note To remark the DSCP, the command flow has highest priority than the command class.

QoS Advanced Configuration

Class Template Presentation

The class templates contain the definitions of advanced QoS parameters which are not settable by the user. The class templates include the definitions of:

- The bits of the DSCP field which are written in the packets,
- The drop policy (Tail drop, RED...),
- Default values for the buffer sizes.

The packet marking in the DSCP field complies with the AF and EF IETF standards. The **101110** pattern is reserved for the EF class.

The following table summarises the DSCP patterns for AF classes. The AF classes are defined by the 3 Most Significant Bits of the DSCP (bold part of the pattern). The 3 Least Significant Bits are the drop precedence bits; they are defined by the *drop precedence* of the **flow** command.

	Class AF1	Class AF2	Class AF3	Class AF4
Low Drop Precedence	00 1010	01 0010	01 1010	10 0010
Medium Drop Precedence	00 1100	01 0100	01 1100	10 0100
High Drop Precedence	00 1110	01 0110	01 1110	10 0110

Drop Policy Presentation

Several drop policies can be used for dropping packets in case of congestion. The drop policies used by **6WIND** predefined class templates are the following:

Taildrop for EF class.

RED for AF classes.

This section gives an overview of the different drop policies implemented in a 6WINDGate.

Taildrop

When using the taildrop policy, incoming packets are dropped when the queue is full.

RED

The Random Early Drop (RED) policy randomly drops or marks packets when the average queue length exceeds a minimum threshold (q_{min}). The drop probability increases with increasing average queue length up to a maximal dropping probability (p_{max}). When the average queue length reaches an upper threshold (q_{max}), all packets are dropped. The average queue length is calculated with an Exponential Weighted Moving Average (EWMA) defined with the averaging parameter P :

$$Q_t = Q_{t-1} + P \cdot (M_t - Q_{t-1})$$

M_t : measured queue length at time t

Q_t : average queue length at time t

P : averaging parameter

The following graph shows how the reject probability evolves.

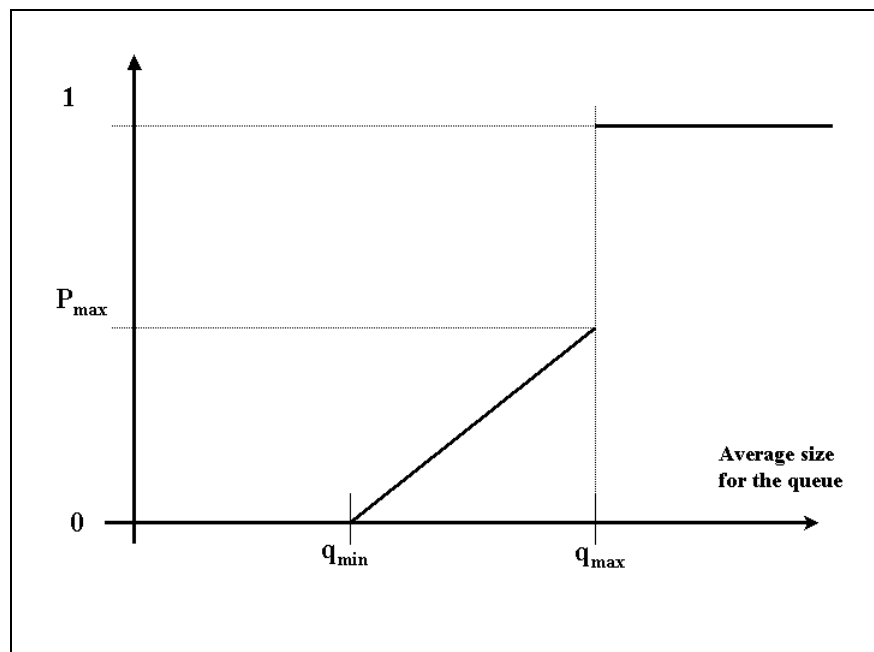


Figure 17: RED algorithm

Configuring a Class Buffer Size

For a dedicated class, a buffer size can be defined using the optional *queue size* parameter of a **class** command.

If the buffer size is not specified, the system will choose a value for it according to the following formulas:

For an EF class: 0.5s of traffic for the specified bandwidth with an upper limitation to 250 kB,

For an AF class: 5s of traffic for the specified bandwidth with an upper limitation to 2 MB.

Configuring the Best Effort Queue Size

The `be_queue` command defines the queue size allowed for the best effort class on an interface:

```
6OS {myconfig-qos} be_queue interface way queue_size unit
```

where:

interface should be set to the logical interface name,

way should be set to direction of the flow (*in* or *out*).

queue_size is set to the allowed queue size for the best effort class on the interface.

unit expresses the bandwidth in *B* (bytes), *KB*, and *MB*.

The default value is 1 MB. The best effort buffer size is set by the user in the qos context.

Example:

```
6OS {myconfig-qos} be_queue eth2_0 out 2 MB
```

Defining the DSCP marking policy for the Best Effort

The packets that not match any classes of services go into the Best Effort Class. The DSCP of these packets can be kept or remarked thanks to the following command:

```
6OS {myconfig-qos} be_dscp interface way remark=value
```

where :

interface should be set to the logical interface name,

way should be set to direction of the flow (*in* or *out*).

value may be the new DSCP value for the interface or **keep_dscp** if the DSCP mark should be unchanged.

The format the value may be binary (%101), decimal (5), hexadecimal (0x05), predefined values: EF, AF11, AF12, AF13, AF21, ... AF43.

The default value is 0.

Example:

```
6OS {myconfig-qos} be_dscp eth2_0 in remark=%111
```

```
6OS {myconfig-qos} be_dscp eth2_0 in remark=AF23
```

```
6OS {myconfig-qos} be_dscp eth2_0 out keep_dscp
```


Configuring the interface in a single line

The QoS for an interface may be configured in a single line:

```
6OS{myconfig-qos} intf interface way bw="bandwidth unit" be_queue="queue_size unit"
keep_dscp|remark=value
```

Where :

interface is set to the logical interface name,

way is set to direction of the flow (*in* or *out*).

bandwidth is set to the maximum allowed bandwidth for the interface. The unit is in bps, Kbps, Mbps.

queue_size is set to the best effort queue size for. The *unit* is *B* (for bytes), *KB* or *MB*.

keep_dscp | **remark**=*value* sets the DSCP remarking policy. **keep-dscp** will leave the DSCP of packets unchanged, as opposed to **remark**, which will remark packets to the provided value. The format for *value* may be binary (%101), decimal (5) or hexadecimal (0x05) or one of the following predefined values: EF, AF11, AF12, AF13, AF21, ... AF43.

Example:

```
6OS{myconfig-qos} intf eth2_0 out bw="2 Mbps" be_queue="100 KB" keep_dscp
```

Configuring the Service Flow

Enable QoS handling for Service Flow

The 6WINDGate SixOS makes it possible to enable or disable Service Flows isolation into a class of service, on each interface available in the device for the outgoing or ingoing traffic.

The Service Flows handling can be enabled or disabled via the following commands:

```
6OS{myconfig-qos} enable sfl interface way
6OS{myconfig-qos} disable sfl interface way
```

where :

interface is the logical interface name,

way is the direction of the flow (*in* or *out*).

Configuring the Service Flows bandwidth

It is possible to define the Service Flow bandwidth via the following command:

```
6OS{myconfig-qos} sfl_bw interface way value unit
```

where:

interface is the logical interface name.

way is the direction of the flow (*in* or *out*).

value is set to the bandwidth reserved to the service flows class.

unit expresses the bandwidth in **bps**, **Kbps**, **Mbps** or as a percentage of the interface maximum bandwidth (**percent** keyword). This expression is case sensitive.

Example:

```
6OS {myconfig-qos} sfl_bw eth2_0 out 1 Mbps
6OS {myconfig-qos} sfl_bw eth2_0 out 2 percent
```

Configuring the Service Flow Queue Size

The `sfl_queue` command defines the queue size of the service flow class on an interface:

```
6OS {myconfig-qos} sfl_queue interface way queue_size unit
```

where:

interface should be set to the logical interface name.

way should be set to direction of the flow (*in* or *out*).

queue_size is set to the queue size for the service flow class on the interface.

unit expresses the bandwidth in *B* (bytes), *KB*, and *MB*.

Example:

```
6OS {myconfig-qos} sfl_queue eth2_0 out 100 KB
```

Defining the DSCP marking policy for the Service Flow

The packets that match the Service Flow Class can keep their DSCP or have it remarked thanks to the following command:

```
6OS {myconfig-qos} sfl_dscp interface way {keep_dscp|remark=value}
```

where :

interface should be set to the logical interface name.

way is the direction of the flow (*in* or *out*).

keep_dscp | **remark=value** sets the DSCP remarking policy. **keep-dscp** will leave the DSCP of packets unchanged, as opposed to **remark**, which will remark packets to the provided value.

The format for *value* may be binary (%101), decimal (5) or hexadecimal (0x05) or one of the following predefined values: EF, AF11, AF12, AF13, AF21, ... AF43.

Example:

```
6OS {myconfig-qos} sfl_dscp in remark=%111
6OS {myconfig-qos} sfl_dscp in remark=AF23
6OS {myconfig-qos} sfl_dscp out keep_dscp
```

Configuring the Service flow class in a single line

The service flow class may be configured in a single line:

```
6OS {myconfig-qos} sfl interface way {enable|disable} bw="bandwidth unit"
queue="queue_size unit" {keep_dscp|remark=value}
```

Where :

interface is the logical interface name,

way should be the direction of the flow (*in* or *out*).

enable or *disable* start or stop QoS handling for service flows,

bandwidth is set to the maximum allowed bandwidth for the interface. *unit* expresses the bandwidth in **bps**, **Kbps**, **Mbps** or as a percentage of the interface maximum bandwidth (**percent** keyword). This expression is case sensitive.

queue_size is set to the queue size for the service flow class on the interface. *unit* expresses the bandwidth in *B* (bytes), *KB*, and *MB*.

keep_dscp | **remark=value** sets the DSCP remarking policy. **keep-dscp** will leave the DSCP of packets unchanged, as opposed to **remark**, which will remark packets to the provided value. The format for *value* may be binary (%101), decimal (5) or hexadecimal (0x05) or one of the following predefined values: EF, AF11, AF12, AF13, AF21, ... AF43.

Example:

```
6OS {myconfig-qos} sfl eth1_0 in enable bw="500 Kbps" queue="120 KB" keep_dscp
```

Monitoring QoS

A private MIB is provided with the SixOS in order to monitor Diffserv QoS functions using SNMP. To get statistics about QoS you'll have to enable `snmp` in the `snmp` context and provide a correct community name, refer to "Configuring SNMP".

6WIND QoS mib is split into three tables which are themselves indexed by tokens:

Classes table:

- The number of the interface it relies on

- The way it is configured (input or output)

- Its number assigned by the SixOS

Flows table

- The number of the interface it relies on

- The way it is configured (input or output)

- Its number assigned by the SixOS

Interfaces table

- The number of the interface

- The way of this interface

For each class the following information are available:

Class Parameters

QosClassInterfaceDescr	The name of the interface corresponding to the interface number (eth0_0, eth1_0, eth2_0)
QosClassName	The name of the class configured by the user.
QosClassPriority	The Template used to configure this class (for example AF1, AF2, AF3, BE)
QosClassDscpRemark	Not implemented yet
QosClassNbBitsDirect	The number of bits processed by this class of service
QosClassNbBitsDiscarded	The number of bits processed by the discard mechanism of this class of service
QosClassNbBitsTrans	The number of bits transmitted by this class of service.
QosClassNbBitsTransOL	The number of bits transmitted over-limit by this class of service.
QosClassNbBitsDroppedAqm	The number of bits dropped by the Algorithmic Queue Management mechanism (TailDrop or RED).
QosClassMaxBandwidth	The maximum bandwidth configured for this class.
QosClassMinBandwidth	The minimum bandwidth guaranteed with this class of service.
QosClassQueueSize	The size of the queue allocated to this class of service.

For each flow the following information are available:

Flow Parameters	
QosFlowInterfaceDescr	The name of the interface corresponding to the interface number.
QosFlowName	The name of the flow configured by the user.
QosFlowClass	The name of the class this flow relies on.
QosFlowDropPrecedence	The drop precedence defined for this flow.
QosFlowSrcAddr	The source address configured for this flow.
QosFlowDstAddr	The destination address configured for this flow.
QosFlowSrcPort	The source port configured for this flow.
QosFlowDstPort	The destination port configured for this flow.
QosFlowProto	The protocol specified for this flow.
QosFlowDscp	The dscp value for this flow.
QosFlowDscpRemark	The dscp remark value for this flow.

For each interface, the following information are available:

Interface Parameters	
QosInterfaceName	The name of the interface corresponding to the interface index.
QosInterfaceState	A boolean value indicating if the interface is QoS enabled or not.

For more help, please refer to the Application Notes “ Monitoring 6Wind equipments with SNMP”.

QoS Configuration Examples

Configuring Expedited Forwarding PHB

Presentation

EF is used to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through a DiffServ domain. Such a service appears to the endpoints like a point-to-point connection or a "virtual based line" (RFC 2598).

The proposed configuration is described Figure 18.

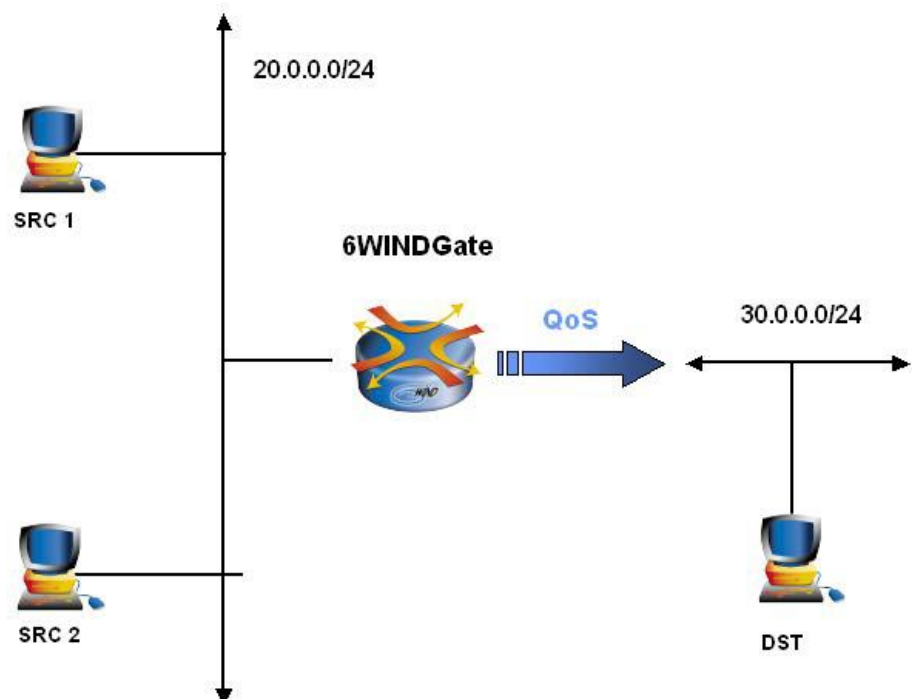


Figure 18: QoS first example

The flows to be transferred over the network are as follows:

A MPEG over UDP / IP traffic of 1.44 Mbps average throughput from SRC1 to DST.

A heavy 4 Mbps IP background traffic generated from SRC2. The bandwidth available is set to 4 Mbps on the outgoing interface of the **6WINDGate** in direction of DST.

6WINDGate Configuration

Step 1 Identification of the QoS requirements

A MPEG traffic of 1.44 Mbps average throughput from SRC1 to DST. A 1.5 Mbps bandwidth is allocated. As the traffic is time sensitive, the necessary bandwidth has to be reserved without any loss.

Step 2 Enabling QoS on the interface

QoS on the outgoing interface is enabled by the following command:

```
6OS{myconfig-eth2_0}enable qos out
```

Step 3 Definition maximum available bandwidth per interface

The maximum allowed bandwidth is 4 Mbps. It is defined by the following command:

```
6OS{myconfig-eth2_0}qos  
6OS{myconfig-qos}maxbandwidth eth2_0 4 Mbps
```

Step 4 Choice of classes

The EF class is chosen because it is adapted to guarantee resources for time-sensitive traffic. The predefined **EF** template is used.

```
6OS{myconfig-qos}class EF_out EF eth2_0 out 1500 Kbps
```

Step 5 Flow Definition and affectation of flows to classes

The MPEG flow is identified by being the unique flow between SRC1 and DST address using the UDP protocol. The other traffic is implicitly affected to the best effort class.

```
6OS{myconfig-qos}flow MPEG_OUT 20.0.0.1/24 30.0.0.0/24 udp EF_out
```

Step 6 Activation of the configuration

Once the configuration has been completed, it can be applied using the `apply conf myconfig` command. The configuration is now the running configuration.

Step 7 Activation at boot time

If the current configuration behaves correctly, make it active at next boot time. The command is:

```
6OS{}copy conf running start
```

Experimental Results

Without any QoS mechanism, all the flows are sent into the best effort class. The MPEG application is heavily disturbed by the background traffic. Experimental results show that only 0.7984 Mbps is available for the MPEG traffic.

Configuring an EF class with 1.5 Mbps for the time - sensitive MPEG flow will make it possible to guarantee the throughput and the jitter.

Furthermore, using the DBS tool provides a graph which displays the jitters on the application level.

The Figure 19 graph shows that when the traffic is sent into the Best Effort class, the jitter climbs up to 0.2 s which is not acceptable for time-sensitive traffic like MPEG. When an EF class is used by the traffic, the jitter is kept low (always under 0.05 s).

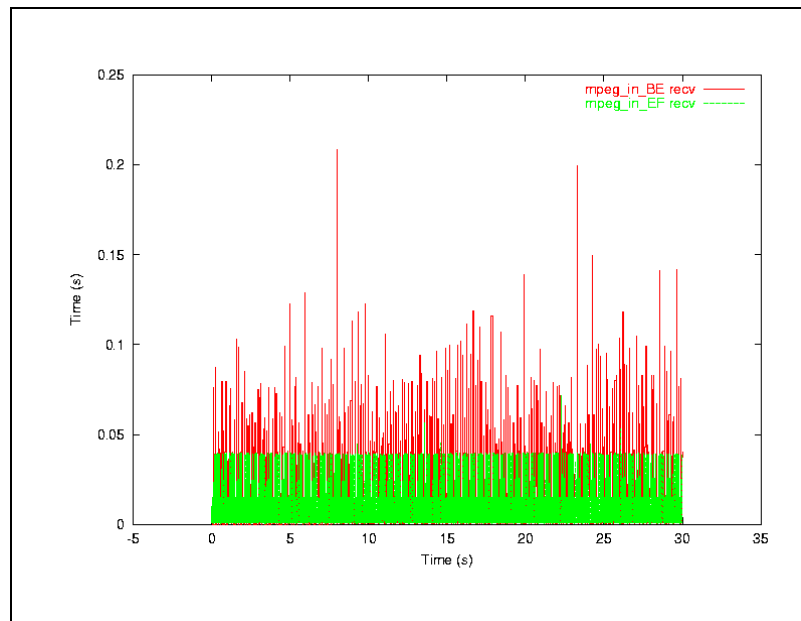


Figure 19: EF experimental results

Configuring Assured Forwarding PHB

Presentation

AF PHB group enables providers to offer packet differentiation. In case of congestion, the drop precedence of a packet determines the relative importance of the packet within the AF class.

6WIND implementation allows defining the drop precedence as a parameter of the filter used for classification. For each drop precedence, a set of RED parameters is defined which allows differentiating certain flows.

AF PHB group enables providers to offer packet differentiation. In case of congestion, the drop precedence of a packet determines the relative importance of the packet within the AF class.

The proposed configuration is described in Figure 20.

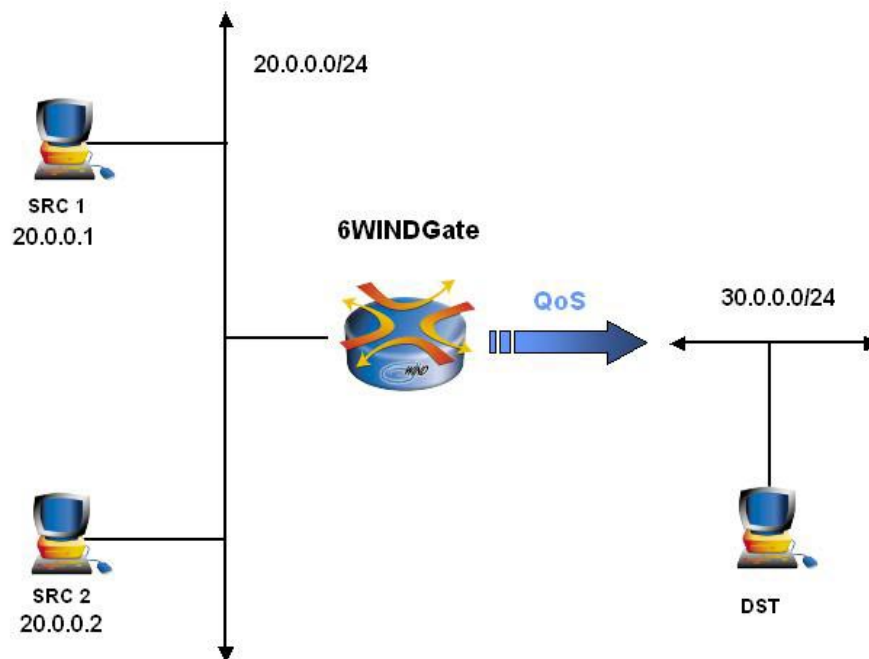


Figure 20: Second QoS example

Two TCP identical traffics, TCP1 and TCP2, are respectively sent from SRC1 to DST and SRC2 to DST, with an increased drop probability for the flow sent by SRC2.

For this test, one AF class is used: AF1 with two levels of drop precedence. The AF1 class has a bandwidth of 1 Mbps and a buffer size of 400 000 bits.

For the TCP1 flow, AF1 is used with a medium drop probability.

For the TCP2 flow, AF1 is used with a high drop probability.

6WINDGate Configuration

Step 1 Identification of the QoS requirements

Two TCP identical traffics are sent from SRC1 to DST and SRC2 to DST, with an increased drop probability for the flow sent by SRC2.

Step 2 Enabling QoS on the interface

QoS on the outgoing interface is enabled by the following command:

```
6OS{myconfig-eth2_0}enable qos out
```

Step 3 Definition maximum available bandwidth per interface

The maximum allowed bandwidth is 4 Mbps. It is defined by the following command:

```
6OS{myconfig-eth2_0}qos
6OS{myconfig-qos}maxbandwidth eth2_0 out 4 Mbps
```

Step 4 Choice of classes

The AF class is chosen because it is adapted to guarantee resources for TCP traffic. The predefined **AF** template is used.

```
6OS{myconfig-qos}class AF1_out AF1 eth2_0 out 1 Mbps 50000 B
```


Step 5 Flow Definition and affectation of flows to classes

```
6OS{myconfig-qos}flow TCP1_OUT 20.0.0.1[50001] 30.0.0.0/24 tcp AF1_out/medium
```

```
6OS{myconfig-qos}flow TCP2_OUT 20.0.0.2[50002] 30.0.0.0/24 tcp AF1_out/high
```

Step 6 Activation of the configuration

Once the configuration has been completed, it can be applied using the `apply myconfig` command. The configuration is now the current configuration.

Step 7 Activation at boot time

If the current configuration behaves correctly, make it active at next boot time. The command is:

```
6OS{}copy running start
```

Experimental results

The AF class has a bandwidth of 1 Mbps and a queue length of 400 000 bits.

The evolution of sequence number for both flows is shown hereafter.

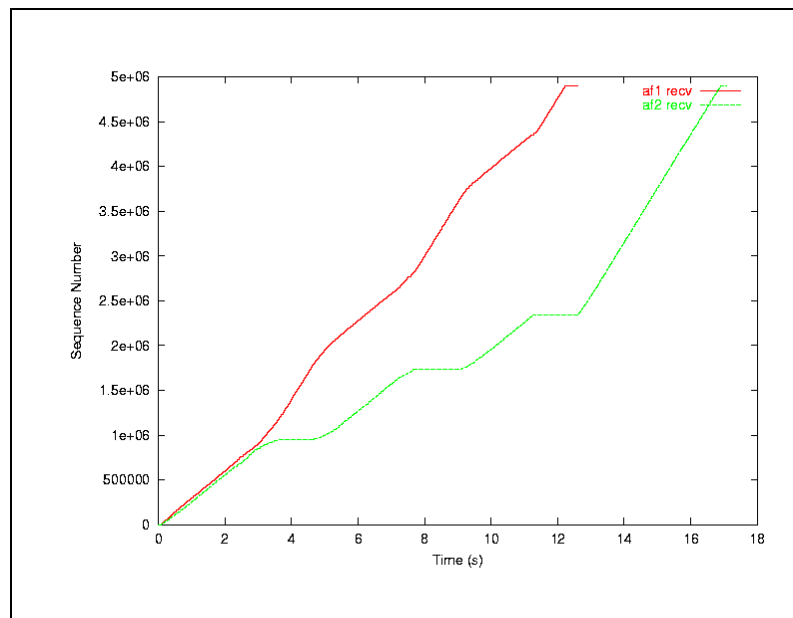


Figure 21: AF experimental results

During the slow-start mechanism (0 to about 3 s), the bandwidth is shared in a fair manner between TCP1 and TCP2 flows. Then, a congestion occurs and RED start dropping more packets from TCP2 due to its higher drop probability.

The average throughput for each flow is the following:

TCP1	3.1242 Mbps
TCP2	2.3037 Mbps

The TCP1 flow was given a much better bandwidth than the TCP2 one.

Chapter 24 Configuring IPv4/v6 IPsec

This chapter describes how to configure IPsec and IKE for IPv4 and IPv6 security management. It also explains how to manage X509 certificates to be used for establishing VPNs with IKE.

Introduction

Configuring the 6WINDGate security mechanisms requires some knowledge of the basic principles of IP security.

Prior reading of the *6WIND security management – White paper* may be helpful.

Definitions

VPN

A VPN (Virtual Private Network) is the association of two IPsec devices acting as security gateways to secure the traffic going from one to another. A VPN defines the general rules to be applied to any traffic transiting from the 6WINDGate to another IPsec device.

Static VPN

A static VPN is a VPN whose security associations are manually defined by the user. This configuration mode can be useful when the use of IKE is not possible or when it is not desired. This could be the case for experimental platforms or when attempting to interoperate with a piece of equipment that does not implement IKE.

Dynamic VPN

A dynamic VPN is a VPN whose security associations are dynamically established using the IKE protocol (Internet Key Exchange). The protocol also ensures dynamic re-keying, which improves the security level.

The establishment of a secure communication is composed of two phases. The first phase (a.k.a. ISAKMP phase) enables IKE peers to authenticate each other and to negotiate cryptographic material (ISAKMP SAs), in order to secure one or more phase 2 exchanges. The second phase (a.k.a. Quick Mode phase) enables to negotiate the IPsec SA themselves. One or more successive phase 2 exchanges are authenticated and encrypted thanks to cryptographic material negotiated during phase 1.

Pre-Shared Key

The use of pre-shared keys is a first option to authenticate IKE peers during phase 1. Pre-shared keys are secret keys shared by two IKE peers to authenticate each other. This key must be loaded in both peers.

This solution does not scale well because when a new device is installed, a new pre-shared key must be generated for each remote device communicating with the new one, and this key must be loaded in both the new and remote devices.

Certificate

The use of certificates is a second option for the IKE authentication process. Certificates used by the 6WINDGate comply with the X509 standard. Certificate-based authentication is a public key cryptographic system.

Each IKE equipment owns a private key and an associated certificate, containing identification information and the equipment public key. The certificate is delivered and signed by a CA (Certification Authority). A CA is an entity identified by a self-signed certificate and owning the associated private key.

IKE peers have decided to trust this CA and believe that the CA will only deliver certificate to authorized equipments. An IKE equipment authenticates an IKE peer by verifying that its certificate was delivered by the CA (thanks to the CA certificate), and that the IKE peer owns the private key associated to its certificate.

When a new device is added into the network, this device simply requests and gets a certificate from the Certificate Authority. No modification of other devices configuration is necessary.

A certificate is valid for finite period of time. When the validity expires, a new certificate must be obtained from the Certification Authority.

VPN Template

A VPN template describes the level of security to be applied on a VPN. It is designed to alleviate the work of configuring VPNs. **6WIND** provides pre-defined templates using pre-shared keys or certificates, but the user can also define its own templates.

IPsec rule

An IPsec rule specifies the IPsec policy that should be applied to an IP flow going from a zone to another zone (basically applying IPsec, letting traffic in clear-text or discarding traffic). When the rule orders to apply IPsec, the IPsec rule is linked to a VPN definition, which will define the involved IPsec devices, the details of the IPsec cryptographic material, and the way it is negotiated.

Several IPsec rules may be linked to the same VPN.

Zone

IP security secures traffic between two IPsec devices (the 6WINDGate and a remote device) that interconnect two sites. Each site may contain several security zones that require different handling in terms of security. A security zone can be a subnetwork or a single host. Two zones are interconnected through a connection secured by IPsec.

Security Association

The IPsec processing that will be applied to a packet is defined by a Security Association (SA). A SA is an agreement between two hosts, that defines notably the IP destination of the communication, the IPsec mechanism to apply (AH or ESP) and its mode, the cryptographic algorithm(s) and key(s), etc. A SA is uniquely identified by the triple (destination address, type of mechanism (AH or ESP), Security Parameters Index (SPI)). The SAs are stored in each machine in a security associations database (SAD). A pair of SA is defined or negotiated for each IPsec rule and IPsec mechanism.

Security Context

The contexts to be used for security configuration are the following:

- The identity parameters are configured in the root context,
- New security templates are defined in the root context,
- Enabling and disabling security is done within a logical interface context,
- All the other commands are set in the `sec` context.

Displaying Security Configuration

Security status on an interface *ifname* can be displayed using the following commands:

```
6OS{myconfig}display ifname
6OS{myconfig-ifname}display
```

where *ifname* indicates the interface name.

To display information about security functions, the following commands can be used:

```
6OS{myconfig-sec}display
6OS{myconfig}display sec
```

Security Configuration Overview

The configuration of the IP security features of 6WINDGates consists in:

- Configuring VPNs,
- Configuring IPsec rules.

Two different levels are available for configuring VPNs:

- Static VPN configuration. In this mode, IKE is not used. The secret keys are provided during configuration and they will not change until a new configuration is issued.
- Dynamic VPN configuration using IKE.

IKE is a protocol developed specifically for IPsec which aims at providing authentication and key exchange mechanisms adapted to most of the situations which can occur on the Internet.

IKE can be used with two authentication methods:

- Use of pre-shared keys,
- Use of certificates.

The configuration of IKE requires to have previously configured the authentication methods to be used.

Use of pre-shared keys

In this mode, a pre-shared key is used. This secret information is used in order to authenticate the two peers of the VPN.

Use of certificates

IKE makes the assumption that the necessary certificates are already loaded on the 6WINDGate.

Several certificates may be available on a single 6WINDGate. This is the case when a device belongs to different communities.

Security Configuration Steps

- Step 1** Identification of the VPNs in which the 6WINDGate takes part (some of the VPNs may be static VPNs). The requirements associated to these VPNs should also be identified.
- Step 2** Enabling security management. It is described in section Enabling Security – Page 142.
- Step 3** If no existing template matches the needs of some of the VPNs, then some new templates should be written. It is described in section Creating a New VPN Template – Page 151. Some pre-defined templates are provided by **6WIND** and it is expected that they will match most of the requirements, so this step would then be unnecessary.
- Step 4** Define the identity parameters necessary for using pre-shared keys or certificates. This definition is described in section Defining Identity Parameters – Page 142.
- Step 5** For each dynamic VPN, install certificates and CA certificates or load the pre-shared keys according to the selected authentication method. This step is described in section Using Certificates - Page 143 for certificates and in section Using Pre-Shared Keys – page 145 for pre-shared keys.
- Step 6** Definition of each VPN. The VPN must reference an existing template that could be pre-defined one or a user-defined one. Defining a VPN is described in section Defining a VPN – Page 145.
- Step 7** For each VPN, identification of the required IPsec rules.
- Step 8** Definition of each IPsec rule, including the IPsec policy selection (ah, esp, ah and esp, discard, clear). Defining an IPsec rule is described in section Defining a Static Security Association– Page 147.
- Step 9** Static VPNs may co-exist with dynamic ones on the same 6WINDGate. If it is the case, the user shall define the Security Associations (SAs) required by the static VPNs. There should be at least one SA per static VPN for outgoing traffic and another one for incoming traffic. The process of adding SAs is described in section Defining a Static Security Association - Page 149.
- Step 10** Apply the configuration to make it the **running** configuration.
- Step 11** If the current configuration behaves correctly, make it active at next boot time.
Section IPsec and IKE Configuration Examples – Page 153, describes step by step configuration examples.

Enabling Security

Security can be enabled or disabled on a per interface basis.

IPsec can be enabled on an interface by using the following command:

```
6OS{myconfig-ifname}enable ipsec
```

IPsec will be disabled on an interface by using the following command:

```
6OS{myconfig-ifname}disable ipsec
```

By default, IPsec is disabled on all the interfaces. If enabled, the `clear` policy is selected by default.

Defining Identity Parameters

Using pre-shared keys or X509 certificates requires defining the 6WINDGate identity. Different identities may be defined on a single 6WINDGate. Each one is defined by a name. The configuration commands for defining identity parameters have to be entered in the root context.

To set up an identity or enter an existing identity context, use the following command:

```
6OS{}id idname
```

Once the command `id` entered, the CLI enters an interactive mode to define the different parameters of the identity. Another way to define an identity is to enter an `id` command with all parameters on a single line.

The parameters to be configured are:

- `idname` is the name for the identity parameters.
- `fqdn` (Fully Qualified Domain Name) is the DNS name that will be used as IKE FQDN identifier and defined as DNS SubjectAltName extension in certificates.
- `user_fqdn` (user Fully Qualified Domain Name) is the email address that will be used as IKE user_FQDN identifier and defined as email SubjectAltName extension in certificates.
- `country`, `state`, `locality`, `organization`, `unit`, `commonname`, and `email` are respectively the C, ST, L, O, OU, CN and E sub-fields of the ASN.1 Distinguished Name that will be used as IKE Distinguished Name identifier and defined as the Subject field in certificates. It is mandatory to define at least one of these fields to use certificates (typically the `commonname` should be defined). It is useless to define the Distinguished Name when only using pre-shared key based VPNs.

If the parameter value includes blank characters, it has to be entered between quotes. The user can assign a parameter value or choose to let this parameter undefined using the `none` value. The `none` value is the default one.

It is mandatory to define the Distinguished Name in order to create certificates, and highly recommended to define at least `fqdn` for use with pre-shared keys.

The list of available identities can be displayed using the following command:

```
6OS{}display id
```

Example:

The following example shows how it can be done using the interactive and non-interactive methods.

```
6OS{}id myidentity 6OS.6wind.net admin.6OS@6wind.fr FR none Paris "6WIND SA"
"Test unit" "Technical Team" test_unit.6OS@6wind.fr

6OS{}id myidentity
enter exit to abort the command
enter fqdn (a string or 'none')[none]: 6OS.6wind.net
enter user_fqdn ( a string or 'none')[none]: admin.6OS@6wind.fr
enter country ( a string or 'none')[none]: FR
enter state ( a string or 'none')[none]:
enter locality ( a string or 'none')[none]: Paris
enter organization ( a string or 'none')[none]: "6WIND SA"
enter unit ( a string or 'none')[none]: "Test unit"
enter commonname ( a string or 'none')[none]: "Technical Team"
enter email ( a string or 'none')[none]: test_unit.6OS@6wind.fr
6OS{}
```

Using Certificates

Introduction

To configure a certificate, the 6WINDGate must interface to a third party PKI. The 6WINDGate uses standard file transfer protocols to send certificate requests and import certificates from the third party certificate generator tool.

Transferring, installing and uninstalling certificates on a device is done at the root level. As 6WINDGates may belong to different communities, several certificates can be installed. Reciprocally, Different configurations may use the same certificate.

At the configuration level, the identity to be used and the CAs to be trusted will be defined. The `vpn` command will specify the right certificate to be used by indicating which CA the VPN depends on.



Caution When using certificates, the user has to pay attention at the validity date of the certificate and the current 6WINDGate date and time configuration.

Installing and Uninstalling CA Certificates and Certificates

The 6WINDGate needs identity information in order to be recognized by the CA and other IPsec devices.

Three identifier types are supported:

- a DNS name (also called FQDN in IKE protocol)
- an email address (also called user_FQDN in IKE protocol)
- an ASN.1 Distinguished Name

The configuration of a new certificate proceeds as follows:

Step 1 First of all, a CA (Certification Authority) must be defined using the following command:

```
6OS{}ca caname ca_URL
```

where:

- *caname* identifies the CA name.
- *ca_URL* defines the CA URL location.

Step 2 Installing a CA certificate is done using the following command:

```
6OS{}import ca_cert caname [cacertremotename]
```

The CA certificate will be loaded using the URL mentioned in the **ca** *caname* command. *cacertremotename* is the CA certificate file to be imported. If the remote file name is omitted, the 6WINDGate imports the *caname.cer* file.

When the command has been successfully executed, a new CA certificate appears in the certificate list managed by the 6WINDGate. This can be verified using the following command:

```
6OS{}display ca_cert
```

Step 3 To get a certificate, the 6WINDGate has to generate a key pair and a certificate request to the CA.

```
6OS{}cert_req idname caname
```

This command generates two files, one including the private key and the second one the certificate request. The certificate request file is named *idname@caname.req*.

The request file must then be transferred to the appropriate CA using an **export** command on the CA remote host.

```
6OS{}export cert_req idname caname [certreqremotename]
```

An **export cert_req** exports a certificate request file to the certification authority previously defined with the **ca** command. If the remote certificate file name is not specified, it will be exported with name *idname.cer*.

The list of available certificate requests can be displayed using the following command:

```
6OS{}display cert_req
```

Step 4 Once the certificate tool has received the 6WINDGate request, it has all the information to generate the certificate. Refer to the tool documentation you are using to generate a certificate at the PEM format.

Step 5 Then, the generated certificate must be loaded on the 6WINDGate. The certificate must be loaded using the command:

```
6OS{}import cert idname caname [certremotename]
```

where:


- *idname* defines the identity name.
- *caname* identifies the CA name.
- *certremotename* is the certificate file to be imported.

When the command has been successfully executed, a new certificate appears in the certificate list managed in the 6WINDGate. This can be verified using the following command:

```
6OS{}display cert
```

Step 6 To make the certificate usable by a configuration, an identity and a trusted CA have to be chosen using the two following commands entered in the **sec** context of a configuration.

```
6OS{myconfig-sec}ike_id idname
6OS{myconfig-sec}trust caname
```

 **Note** Be sure that the CA certificate corresponding to the trusted CA name has been imported (cf. step 2).

Using Pre-Shared Keys

A pre-shared key is a secret key shared by two IKE peers. Before using pre-shared keys, you have to define them and to store them on the 6WINDGate.

Pre-shared keys can be added using the following command:

```
6OS{myconfig-sec}psk remote_identifier key
```

where:

- *remote_identifier* is the id of the remote IPsec peer that shares this pre-shared key. It may be of the following types :
 - IPv4 or IPv6 addresses,
 - email addresses (user_FQDNs),
 - DNS names (FQDNs).
- *key* is the common pre-shared key. It is a byte string. For security reasons, it is advised to choose a string length of at least 20 bytes. It can be typed as a character string of at least 20 characters or as a hexadecimal string of at least 40 characters (0123456789abcdef) with a leading "0x". The key does not accept blank characters and quotes.

The type of ID is defined in the *ike_type* parameter of the VPN template file. The value of the ID is defined in the remote IPsec peer.

Example:

For instance, the 6WINDGate IP6OS, identified by FQDN *hurricane.6wind.com*, will perform a pre-shared key based IKE negotiation with IPED2 identified by IPv6 address:

```
3ffe:327:450:ffff:215:3fff:fea2:8c56
```

To do so, the following command has to be used in the IP6OS:

```
6OS{myconfig-sec}psk 3ffe:327:450:ffff:215:3fff:fea2:8c56
gonewiththe6windforthenewinternet
```

On the other hand, IPED2 will be programmed with the following command:

```
ED2{myconfig2-sec}psk hurricane.6wind.com gonewiththe6windforthenewinternet
```

A pre-shared key can be deleted using the following command:

```
6OS{myconfig-sec}delete psk identifier
```

To delete all the defined pre-shared keys, use the following command:

```
6OS{myconfig-sec}delete psk all
```

Defining a VPN

To define a VPN, the following command is used.

```
6OS{myconfig-sec}vpn vpnname templatename endpoint1 endpoint2 [caname]
```

where:

- *vpnname* is the name chosen for the VPN.

- `vpntemplate` should be set to the name of the template to be used by the VPN. The template should be selected in the following list:
 - o `static`: static configuration of IPsec. IKE is not used.
 - o `psk_strong`: stronger security with IKE and Pre-Shared Key authentication,
 - o `psk_lite`: lighter security but with lower overhead and Pre-Shared Key authentication,
 - o `cer_strong`: stronger security with IKE and certificates based authentication,
 - o `cer_lite`: lighter security but with lower overhead and certificates based authentication,
 - o `xxxx`: optional user defined template defined using a `vpntemplate` command. Refer to the Creating a New VPN Template section – Page 151 to define your own template.

The following table summarizes the IKE parameters selected for the pre-defined dynamic VPN templates.

<i>name</i>	<i>cer_strong</i>	<i>cer_lite</i>	<i>psk_strong</i>	<i>psk_lite</i>
<i>mode</i>	main	aggressive	main	aggressive
<i>phase1_crypt</i>	3des	des	3des	des
<i>phase1_hash</i>	sha1	md5	sha1	md5
<i>phase1_time</i>	1 hour	16 hour	1 hour	16 hour
<i>phase1_bytes</i>	6000 KB	60000 KB	6000 KB	60000 KB
<i>dh_group</i>	2	1	2	1
<i>auth_method</i>	rsasig	rsasig	pre_shared_key	pre_shared_key
<i>id_type</i>	asn1dn	asn1dn	address	fqdn
<i>phase2_crypt</i>	3des	des	3des	Des
<i>phase2_auth</i>	hmac_sha1	hmac_md5	hmac_sha1	hmac_md5
<i>phase2_bytes</i>	1000 MB	2000 MB	1000 MB	2000 MB
<i>phase2_time</i>	1 hour	8 hour	1 hour	8 hour
<i>pfs_group</i>	1	1	1	1



Caution When using main mode and the pre-shared key authentication method, only address identifier type is supported.

Refer to the *6WINDGate SixOS - Command Reference* to obtain details about IKE parameters selected for **6WIND** pre-defined templates.

- `endpoint1` should be set to the IP address of the local end-point of the VPN. IP addresses may be IPv4 or IPv6 ones but `endpoint1` and `endpoint2` must be of same IP version.
- `endpoint2` should be set to the IP of the remote end-point of the VPN.
- `caname` is an optional parameter that must be set to the name of the certification authority if a certificate based authentication is used. A certificate matching the IKE ID and delivered by this certification authority must have been installed on the 6WINDGate. If the VPN does not use certificate authentication, then this parameter will be ignored.

An existing VPN can be deleted using the following command:

```
6OS{myconfig-sec}delete vpn vpnname
```

All defined VPNs can also be deleted using the following command:

```
6OS{myconfig-sec}delete vpn all
```

Example:

```
6OS{myconfig-sec}vpn vpn1 cert_lite 192.0.0.1 192.0.0.2 cacommunity1
                                # IPv4 using a certificate
6OS{myconfig-sec}vpn vpn2 psk_lite 3ffe::1 3ffe::2
                                # IPv6 using pre-shared keys

6OS{myconfig-sec}delete vpn vpn1
```

Defining a roadwarrior VPN

A roadwarrior VPN is a special VPN that handles remote IKE devices whose IP address is unknown. This kind of VPN is typically used to handle traveling hosts connecting to the internet from different access points, and dynamically acquiring temporary IP addresses. The IKE connection is initiated by the remote host (a.k.a. roadwarrior). Only one roadwarrior VPN may be defined.

```
6OS{myconfig-sec}vpn vpnname templatename roadwarrior [caname]
```

Parameters of a roadwarrior VPN are the same as a standard VPN, except IP addresses, which are omitted. The roadwarrior VPN handles both IPv4 and IPv6 IKE negotiations. Only transport mode ipsec-rules may be linked to the roadwarrior VPN.

Note The roadwarrior functionality is an advanced function. The way it is configured may be changed without prior notice.

Defining an IPsec rule

An IPsec rule specifies the IPsec policy that should be applied to an IP flow going from a zone to another zone. It is composed of two part: a filter that defines a traffic flow, and an action, which describes what IPsec processing the traffic flow should experience.

As traffic flow definitions may overlap, ipsec-rules are ordered, through a priority value. If a packets matches two or more ipsec-rules, the rule with the smallest priority will be chosen. Two ipsec-rules with same priority should consequently not overlap, to avoid ambiguities.

Here is the syntax for the ipsec-rule command:

```
6OS{myconfig-sec}ipsec-rule name localzone remotezone proto [via ifname]
ipsecpolicy mode vpnname [priority]
6OS{myconfig-sec}ipsec-rule name localzone remotezone proto [via ifname]
otherpolicy [priority]
```

where:

The first part of the command is the traffic flow definition:

- *name* is the name of the ipsec-rule
- *localzone* describes the local zone of the traffic flow in the form of an IP address range. It must be set as IPv4/v6 address or IPv4/v6 address range. It may be completed by TCP/UDP port specification for static VPNs only. The following forms are allowed:
 - *address*
 - *address/prefixlen*
 - *address[port]*
 - *address/prefixlen[port]*

- o (*prefixlen* and *port* must be decimal numbers. The square bracket around *port* is really necessary. No name resolution is performed on addresses. The addresses must be specified in numeric form)
- *remotezone* describes the remote zone of the traffic flow. Format is the same as *localzone*.
- *proto* is the upper protocol carried by IP. It may be *any*, *udp*, *tcp*, *icmp*, *icmp6* or a numeric protocol value. If *icmp* or *icmp6* is used, the ICMP type value may also be specified, in square brackets (e.g. *icmp*[8]=“icmp echo request”, *icmp6*[1]=“destination unreachable”).
- *via ifname* is an optional interface name. The *ipsec-rule* will only be defined on the specified interface. This option can only be used with static VPNs.

The second part of the command is the action definition. If IPsec must be applied to the traffic flow, the first form of the command is used:

- *ipsecpolicy* is the IPsec mechanisms that will be applied
 - o **ah** requires the use of AH
 - o **esp** requires the use of ESP
 - o **esp_ah** requires the use of ESP and AH
 - o
 - o **discard** if the tunnel requires packet discard (traffic is forbidden)
 - o **clear** if the tunnel requires no IPsec handling. (such tunnel definition is useless, since it is the default IPsec policy)
- *mode* is the mode of the IPsec mechanism (**tunnel** or **transport**).
- *vpnname* is the name of the VPN that this rule is linked to. The VPN will provide information about the IPsec gateways and the cryptographic material. In case of a transport *ipsec-rule* and of manual configuration (i.e. without IKE), the *vpnname* may be set to **static**. Manual configuration of IPsec SAs will nevertheless remain necessary.

If IPsec must not be applied to the traffic flow, the second form of the command is used:

- *otherpolicy* the action that will be applied
 - o **clear** let the traffic flow pass in clear-text
 - o **discard** discard packets of the traffic flow

Eventually, the priority of the *ipsec-rule* may be configured to avoid rule overlaps.

- *priority* the priority of the rule. Smaller priorities are “preferred” rules.

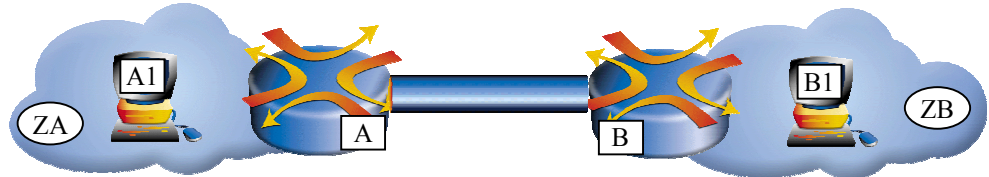


Caution The user will pay attention not to let rules with same priority overlap. Priority is used to avoid ambiguities.



Caution Use of transport mode must be accurately understood. Even if an ipsec-rule may specify localzone and remotezone as a subnet, IKE negotiation and IPsec are performed between the two endpoints located in each zone, not between two security gateways. The localzone must only contain addresses configured on the 6WINDGate. And the VPN definition must be configured accurately.

For instance, the following configuration is invalid:



```
ipsec-rule ZA ZB any esp transport VPNAB
ipsec-rule A B any esp transport VPNAB
vpn VPNAB cer_strong A B CANAME
trust CANAME
ike_id routerA
```

An existing ipsec-rule can be deleted using the following command:

```
6OS{myconfig-sec}delete ipsec-rule name
```

All ipsec-rules can be deleted using the following command:

```
6OS{myconfig-sec}delete ipsec-rule all
```

Example:

```
6OS{myconfig-sec}ipsec-rule ZAZB 10.0.0.0/24 10.1.0.0/24 tcp ah tunnel vpn4
6OS{myconfig-sec}ipsec-rule ABicmp 192.0.0.1 192.0.0.2 icmp clear 2009
6OS{myconfig-sec}ipsec-rule AB 192.0.0.1 192.0.0.2 any ah transport vpn4 2010
```

When a tunnel applies to any traffic, the `0.0.0.0/0` or `::/0` notation is used. Defining a clear policy between a zone and the Internet can be done as follows:

```
6OS{myconfig-sec}ipsec-rule freeexit 10.0.0.0/24 0.0.0.0/0 any clear
```

```
6OS{myconfig-sec}delete ipsec-rule rule1
```



Note When protecting traffic between two gateways by a static VPN, the same static SAs will be used for all ipsec-rules attached to this VPN. When using a dynamic VPN, IKE will negotiate different SAs for each tunnel where traffic flows.

Security Advanced Configuration

Defining a Static Security Association

When static VPNs are used, the user must manually specify the SA required by the static VPNs.

There should be at least one SA (security association) per static VPN for outgoing traffic and another one for incoming traffic.

Specifying an SA in the configuration file can be done using the following commands:

```
6OS{myconfig-sec}sa_ah srcaddress dstaddress spi algo key
6OS{myconfig-sec}sa_esp srcaddress dstaddress spi algo key [aalgo akey]
```

The first command adds a static SA for AH; the second adds a static SA for ESP. If both AH and ESP are required, two commands are required.

The command arguments are :

- *srcaddress/dstaddress* Source/destination of the secure communication (IPv4 or IPv6 addresses). The commands do not perform name resolution for the *srcaddress* and *dstaddress* arguments. They must be in numeric form.
- *spi* Security Parameter Index (SPI) for the SA. It must be a decimal number or a hexadecimal number (with leading 0x). The range of SPI values between 0 and 255 cannot be used. A SPI value can be shared between several SAs, provided their triples (destination address, type of mechanism (AH or ESP), Security Parameters Index (SPI)) differ.
- *algo* The cryptographic algorithm specification. The following list shows the algorithms that are supported. Protocols and algorithms are almost orthogonal.
- *key* Key for the chosen algorithm. The key length are specified in the following table. This parameter must be a hexadecimal value or a string (specified between double quotes).
- *aalgo* The optional authentication algorithm specification for ESP mechanism. The following list shows the algorithms that are supported. Protocols and algorithms are almost orthogonal.

authentication algorithm	key length (bits)	comments
hmac-md5	128	rfc2403
hmac-sha1	160	rfc2404

The supported ESP algorithms are listed in the following table.

encryption algorithm	key length (bits)	comments
des-cbc	64	rfc2405
3des-cbc	192	rfc2451
aes-cbc	128/192/256	rfc 3394

Note Some algorithms may or may not be available according to the local regulation.

An existing static security association can be deleted using the following command:

```
6OS{myconfig-sec}delete sa_ah dstaddress spi
6OS{myconfig-sec}delete sa_esp dstaddress spi
```

All the existing static security associations can also be deleted using the following command:

```
6OS{myconfig-sec}delete sa_ah all
```

```
6OS{myconfig-sec}delete sa_esp all
```

Example:

```
6OS{myconfig-sec}sa_ah 20.0.0.1 30.0.0.1 11000 hmac-md5
0x000102030405060708090A0B0C0D0E0F
6OS{myconfig-sec}sa_ah 3ABC::1 3DEF::1 11004 hmac-sha1 "keykeykeykeykeykeykeyke"
6OS{myconfig-sec}sa_esp 20.0.0.1 30.0.0.1 11001 des-cbc 0x0001020304050607
6OS{myconfig-sec}sa_esp 3ABC::1 3DEF::1 11004 3des-cbc "keykeykeykeykeykeykeykeykey"
6OS{myconfig-sec}sa_esp 3ABC::1 3DEF::1 11005 3des-cbc "keykeykeykeykeykeykeykeykey"
hmac-sha1 "keykeykeykeykeykeykeyke"
6OS{myconfig-sec}sa_esp 10.23.4.104 10.23.4.205 12345679 aes-cbc "128Bits-
16Bytes-" hmac-sha1 "160Bits-20Bytes-Key-"
6OS{myconfig-sec}sa_esp 10.23.4.104 10.23.4.205 12345680 aes-cbc "192Bits-
24Bytes-Key-Key-Key-" hmac-sha1 "160Bits-20Bytes-Key-"
6OS{myconfig-sec}sa_esp 10.23.4.104 10.23.4.205 12345681 aes-cbc "256Bits-
32Bytes-Key-Key-Key-Key-" hmac-sha1 "160Bits-20Bytes-Key-"

6OS{myconfig-sec}delete sa_esp 3DEF::1 11004
```

Creating a New VPN Template

A VPN template describes a level of security to be applied to a VPN.

6WIND provides pre-defined templates, but the user can also define other templates if needed. **6WIND** pre-defined templates cannot be modified. However, it must be noted that designing new templates is a difficult task that requires a lot of security expertise. A bad configuration may lead to either security leaks, either to very big resource consumptions that would lead to performance collapse.

New security templates have to be defined at the root context. The new templates will be defined thanks to the following command:

```
6OS{myconfig}vpntemplate vpntemplatename
```

Once the command **vpntemplate** entered, the CLI enters an interactive mode to define the 13 parameters of the template. Another way to define a security template is to enter a **vpntemplate** command with all the parameters on a unique line.

The following example shows how it can be done using both methods.

```
6OS{ }vpntemplate myvpntemplate # enters interactive mode
enter exit to abort the command

ike_enable (enable|disable)[enable]:
enter ike_mode (main|aggressive|<list>)[main]: aggressive
enter phase1_crypt (des|3des)[3des]: des
enter phase1_hash (md5|sha1)[sha1]: md5
enter phase1_time ("value hour|min|sec")["1 hour"]:"16 hour"
enter phase1_bytes ("value B|KB|MB")["6000 KB"]:"60000 KB"
enter dh_group (1|2|5)[2]:
enter auth_method (pre_shared_key|rsasig)[rsasig]:
enter id_type (fqdn|address|user_fqdn|asn1dn)[asn1dn]:
enter phase2_crypt (aes|aes192|aes256|3des|des|null_enc|<list>)[aes]: 3des,des
enter phase2_auth (hmac_md5|hmac_sha1|non_auth|<list>)[hmac_sha1]:
non_auth,hmac_sha1
enter phase2_time ("value hour|min|sec")["1 hour"]:"8 hour"
```

```

enter phase2_bytes ("value B|KB|MB") ["5000 MB"] : "500000 MB"
enter pfs_group (1|2|5|none) [none] :
6OS{

6OS{vpntemplate myvpntemplate enable aggressive des md5 "16 hour" "60000 KB" 2
rsasig asn1dn 3des,des non_auth,hmac_sha1 "8 hour" "500000 MB" none
6OS{

```

The significance of the different parameters is as follows. The accepted values for each parameter are listed between parentheses.

- *ike_enable* set to *enable* means that IKE should be used. When set to *disable*, it means that the VPN is static: it does not use IKE. In this case, the rest of the file is useless. The template corresponding to static VPN already exists and is named **static**.
- *ike_mode* defines the IKE phase-1 mode to use (*aggressive* or *main* or *list*).
- *phase1_crypt* is the encryption algorithm to use in IKE phase 1 (*des* or *3des*).
- *phase1_hash* is the hash algorithm to use in IKE phase 1 (*md5* or *sha1*).
- *phase1_time* is the ISAKMP SAs timeout in units of time.
- *phase1_bytes* is the ISAKMP SAs timeout in units of bytes.
- *dh_group* is the Diffie Hellman Group that will be used in phase 1 (*1* (for MODP768), *2* (for MODP1024) or *5* (for MODP1536)).
- *auth_method* is the method used to authenticate peers in phase 1 (*pre_shared_key* or *rsasig* for certificate-based authentication).
- *id_type* is the identifier type that the 6WINDGate will use to identify itself in phase 1: *address* for an IPv4/v6 address, *fqdn* for DNS name, *user_fqdn* for email address or *asn1dn* for ASN.1 Distinguished Name. *asn1dn* can only be used with certificate-based authentication (*auth_method = rsasig*). *address* can only be used with a pre-shared key based authentication (*auth_method = pre_shared_key*).
- *phase2_crypt* is the list of encryption algorithms that will be proposed in phase 2 (*des*, *3des*, *aes*, *aes192*, *aes256* or a comma-separated list) for ESP mechanism.
- *phase2_auth* is the list of authentication algorithms that will be proposed in phase 2 (*hmac_md5* or *hmac_sha1* or *no_auth* or a list) for AH mechanism and for the authentication part of the ESP mechanism.
- *phase2_time* is the IPsec SAs timeout in units of time.
- *phase2_byte* is the IPsec SAs timeout in units of bytes.
- *pfs_group* is the Diffie Hellman Group that will be used in phase 2 if Perfect Forward Secrecy is enabled. (*1* (for MODP768), *2* (for MODP1024) or *5* (for MODP1536), or *none* to disable PFS).

Refer to the *6WINDGate SixOS - Command Reference* to obtain details about these different parameters.

Deleting a User Defined VPN Template

A user-defined VPN template can be deleted using the following command:

```
6OS{myconfig}delete vpntemplate vpntemplatename
```


IPsec and IKE Configuration Examples

Presentation

Three examples of configuration of the IP security features of 6WINDGates are provided here:

- Configuration of a static VPN,
- Configuration of an IKE VPN with pre-shared key authentication,
- Configuration of an IKE VPN with certificate authentication.

The three examples all use the same scenario which is described only once at the beginning of this chapter.

Scenario

Let's consider the Figure 22 – Security configuration example where “Gateway 1” must be configured:

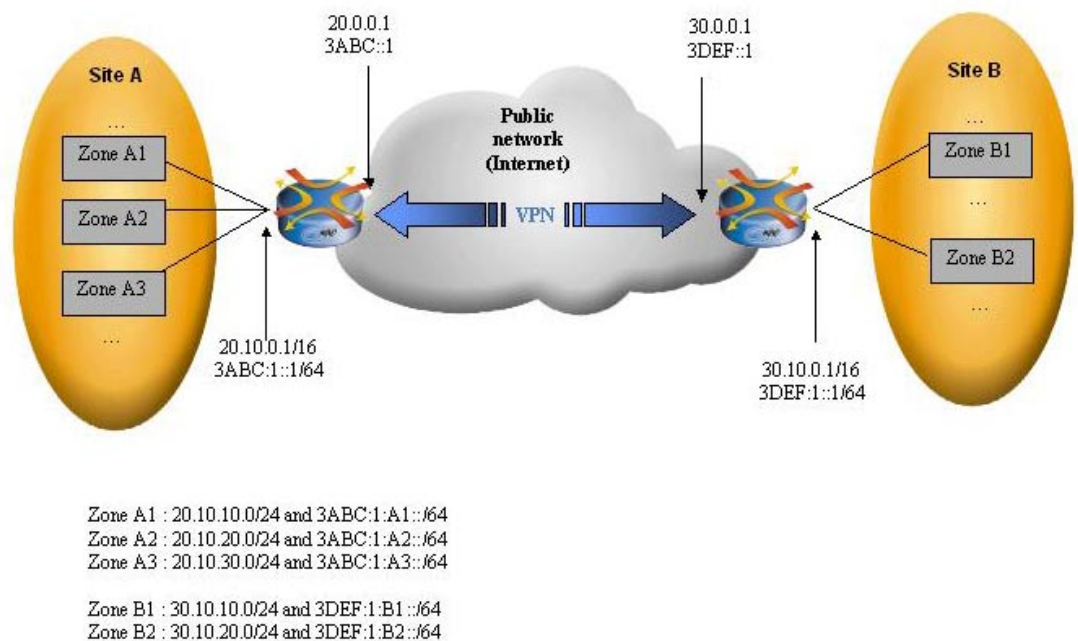


Figure 22 – Security configuration example

Here, 6WINDGates connect two sites to the Internet (via IPv4 and IPv6) and they must secure traffic between the two sites. The requirements are as follows:

- IPv4 traffic between zones A1 and B1 should be protected by AH (authentication)
- IPv6 traffic between zones A2 and B2 should be protected by AH + ESP (authentication and encryption)
- All IPv4 and IPv6 traffic between the two 6WINDGates must be protected by ESP in transport mode
- No traffic (IPv4 and IPv6) is allowed between A1 and B2.
- IPv4 and IPv6 traffic between A1 and the rest of the Internet should be allowed.
- No traffic (IPv4 and IPv6) is allowed between A3 and outside site A.

In the example described above, Gateway 1 has one VPN set with Gateway 2 and this VPN includes 8 ipsec-rules:

- ipsec-rule 1: A1 ⇔ B1 with AH mandatory for IPv4
- ipsec-rule 2: P1 ⇔ P2 with ESP mandatory for IPv4
- ipsec-rule 3: P1 ⇔ P2 with ESP mandatory for IPv6
- ipsec-rule 4: A1 ⇔ B2 with no traffic allowed for IPv4
- ipsec-rule 5: A1 ⇔ Internet with any traffic allowed for IPv4
- ipsec-rule 6: A2 ⇔ B2 with AH and ESP mandatory for IPv6
- ipsec-rule 7: A1 ⇔ B2 with no traffic allowed for IPv6
- ipsec-rule 8: A1 ⇔ Internet with any traffic allowed for IPv6

In the following examples, *myconfig* is assumed to be the configuration under definition. The context under which the commands have to be entered is indicated in the different command prompts.

Configuring a Static VPN

Step 1 to 3 Identification of the requirements – Choice of a template – Security enabling

In this case, the template to use is the `static` one. Security has to be enabled on the public interface.

Step 4 Definition of identity parameters

When only static VPNs are used on the device, these parameters are not required.

Step 5 Pre-shared keys and certificates

This step is not needed in the case of static VPN.

Step 6 VPN Definition

The following commands define a first VPN called `static_vpn4` for IPv4 traffic and a second one called `static_vpn6` for IPv6 traffic.

```
6OS{myconfig-sec}vpn static_vpn4 static 20.0.0.1 30.0.0.1
6OS{myconfig-sec}vpn static_vpn6 static 3abc::1 3def::1
```

The template used is the `static` one.

The IPv4 and IPv6 addresses of the VPN end-points are defined (P1 is the local gateway and P2 is the remote one).

`caname` is not needed in such a static VPN.

Step 7 and 8 IPsec rules Definition

In this example, there are 8 IPsec rules to define.

The first step consists in sorting the ipsec-rules so that overlapping definitions are correctly ordered. For instance, ipsec-rule 5 must have a lower priority than ipsec-rule 4 (and the same for rules 8 and 7).

ipsec-rule 1: A1 ⇔ B1 with AH mandatory for IPv4

```
6OS{myconfig-sec}ipsec-rule rule1 20.10.10.0/24 30.10.10.0/24 any ah tunnel
static_vpn4
```

ipsec-rule 2: P1 ⇔ P2 with ESP mandatory for IPv4

```
6OS{myconfig-sec}ipsec-rule rule2 20.0.0.1 30.0.0.1 any esp transport static_vpn4
2000
```

ipsec-rule 3: P1 ⇔ P2 with ESP mandatory for IPv6

```
6OS{myconfig-sec}ipsec-rule rule3 3abc::1 3def::1 any esp transport static_vpn4
2000
```

ipsec-rule 4: A1 ⇔ B2 with no traffic allowed for IPv4

```
6OS{myconfig-sec}ipsec-rule rule4 20.10.10.0/24 30.10.20.0/24 any discard 2000
```

ipsec-rule 5: A1 ⇔ Internet with any traffic allowed in clear for IPv4

```
6OS{myconfig-sec}ipsec-rule rule5 20.10.10.0/24 0.0.0.0/0 any discard 2100
```

ipsec-rule 6: A2 ⇔ B2 with AH and ESP mandatory for IPv6

```
6OS{myconfig-sec}ipsec-rule rule6 3abc:1:a2::/64 3def:1:b2::/64 any esp_ah tunnel
static_vpn4 2000
```

ipsec-rule 7: A1 ⇔ B2 with no traffic allowed for IPv6

```
6OS{myconfig-sec}ipsec-rule rule7 3abc:1:a1::/64 3def:1:b2::/64 any discard 2000
```

ipsec-rule 8: A1 ⇔ Internet with any traffic allowed for IPv6

```
6OS{myconfig-sec}ipsec-rule rule8 3abc:1:a1::/64 ::/0 any discard 3000
```

Step 9 Manual addition of SAs

When static VPNs are used, the user must specify the security associations required by the static VPNs.

In order to add the SAs required, the following commands have to be entered:

```
# SA definitions AH and ESP in both directions between the
# two gateways - Definition of the VPN.
```

```
6OS{myconfig-sec}sa_ah 20.0.0.1 30.0.0.1 1100 hmac-md5
0x000102030405060708090A0B0C0D0E0F
6OS{myconfig-sec}sa_ah 30.0.0.1 20.0.0.1 10001 hmac-md5
0x0A0B0C0D0E0F00010203040506070809
6OS{myconfig-sec}sa_ah 3ABC::1 3DEF::1 11003 hmac-md5
0x060708090A0B0C0D0E0F000102030405
6OS{myconfig-sec}sa_ah 3DEF::1 3ABC::1 10003 hmac-md5
0x0001020304050C0D0E0F060708090A0B
6OS{myconfig-sec}sa_esp 3ABC::1 3DEF::1 11004 3des-cbc "keykeykeykeykeykeykeykeykey"
6OS{myconfig-sec}sa_esp 3DEF::1 3ABC::1 10004 3des-cbc "eykeykeykeykeykeykeykeykeyk"
```

Step 10 Activation of the configuration

Once the configuration has been completed, it can be applied using the `apply conf myconfig` command. The configuration is now the current configuration.

Step 11 Activation at boot time

If the current configuration behaves correctly, make it active at next boot time. The command is:

```
6OS{}copy conf running start
```

Configuring a Dynamic VPN with Pre-Shared Key Authentication

Step 1 to 3 Identification of the requirements – Choice of a template – Security enabling

In this case, the template to use could be the `psk_lite` one or the `psk_strong` one. Assuming the example requires strong security, let's select the `psk_strong` template. Security has to be enabled on the required interface.

Step 4 Definition of identity parameters

To use pre-shared keys, the user must define the 6WINDGate identity parameters using the following command:

```
# subjectAltNames
6OS{}id ident1 6OS.6wind.com ip6OS@6wind.com none none none none none none none
```

Defining X509 Distinguished Name is not necessary when using pre-shared keys.

Step 5 Pre-shared keys and certificates

The identity used for the VPN has to be defined using the following command:

```
6OS{myconfig-sec}ike_id ident1
```

The pre-shared keys must be defined. Gateway 1 performs a pre-shared key based IKE negotiation with Gateway 2 identified by its IP address (30.0.0.1) then the following command can be used:

```
6OS{myconfig-sec}psk 30.0.0.1 gonewiththe6wind
# gonewiththe6wind is the pre-shared key with peer 30.0.0.1
```

On Gateway 2, a `psk` command will also be defined. For Gateway 2, Gateway 1 will also be identified by its IP address (20.0.0.1).

```
ED2{my2config-sec}psk 20.0.0.1 gonewiththe6wind
```

Step 6 VPN Definition

The following commands define a first VPN called `psk1_vpn4` for IPv4 traffic and a second one called `psk1_vpn6` for IPv6 traffic.

```
6OS{myconfig-sec}vpn psk1_vpn4 psk_strong 20.0.0.1 30.0.0.1
6OS{myconfig-sec}vpn psk1_vpn6 psk_strong 3abc::1 3def::1
```

The template used is the `psk_strong`.

The IPv4 and IPv6 addresses of the VPN endpoints are defined (P1 is the local gateway and P2 is the remote one).

Step 7 and 8 IPsec rules Definition

The IPsec rules are the same as for a manual (static) configuration.

ipsec-rule 1: A1 ⇔ B1 with AH mandatory for IPv4

```
6OS{myconfig-sec}ipsec-rule rule1 20.10.10.0/24 30.10.10.0/24 any ah tunnel
static_vpn4
```

ipsec-rule 2: P1 ⇔ P2 with ESP mandatory for IPv4

```
6OS{myconfig-sec}ipsec-rule rule2 20.0.0.1 30.0.0.1 any esp transport static_vpn4
2000
```

ipsec-rule 3: P1 ⇔ P2 with ESP mandatory for IPv6

```
6OS{myconfig-sec}ipsec-rule rule3 3abc::1 3def::1 any esp transport static_vpn4
2000
```

ipsec-rule 4: A1 ⇔ B2 with no traffic allowed for IPv4

```
6OS{myconfig-sec}ipsec-rule rule4 20.10.10.0/24 30.10.20.0/24 any discard 2000
```

ipsec-rule 5: A1 ⇔ Internet with any traffic allowed in clear for IPv4

```
6OS{myconfig-sec}ipsec-rule rule5 20.10.10.0/24 0.0.0.0/0 any discard 2100
```

ipsec-rule 6: A2 ⇔ B2 with AH and ESP mandatory for IPv6

```
6OS{myconfig-sec}ipsec-rule rule6 3abc:1:a2::/64 3def:1:b2::/64 any esp_ah tunnel
static_vpn4 2000
```

ipsec-rule 7: A1 ⇔ B2 with no traffic allowed for IPv6

```
6OS{myconfig-sec}ipsec-rule rule7 3abc:1:a1::/64 3def:1:b2::/64 any discard 2000
```

ipsec-rule 8: A1 ⇔ Internet with any traffic allowed for IPv6

```
6OS{myconfig-sec}ipsec-rule rule8 3abc:1:a1::/64 ::/0 any discard 3000
```

Step 9 Manual addition of SAs

As the VPN is not static, no manual addition of SAs is required. SAs will be dynamically negotiated, via the IKE protocol.

Step 10 Activation of the configuration

Once the configuration has been completed, it can be applied using the `apply conf myconfig` command. The configuration is now the current configuration.

Step 11 Activation at boot time

If the current configuration behaves correctly, make it active at next boot time. The command is:

```
6OS{}copy conf running start
```

Configuring a Dynamic VPN with Certificate Authentication

Step 1 to 3 Identification of the requirements – Choice of a template – Security enabling

In this case, the template to use could be the `cer_lite` one, or the `cer_strong` one. Assuming the example requires strong security, let's select the `cer_strong` template. Security has to be enabled on the required interface.

Step 4 Definition of identity parameters

To use certificates, the user must define the 6WINDGate identity parameters using the following command:

```
6OS{id ident1 6OS.6wind.com ip6OS@6wind.com US California "Mountain View" 6WIND
none "6WINDGate1" ip6OS@6wind.com
```

In order to set up the Certification Authority, enter:

```
6OS{ca allthewinds scp://ca.6wind.com/ca
6OS{import ca_cert allthewinds [remotename]
```

In order to set up the 6WINDGate certificate, enter:

```
6OS{cert_req ident1 allthewinds
6OS{export cert_req ident1 allthewinds
```

To generate a certificate with a third-party tool, please refer to the tool documentation. Normally the tools creates the certificate:

```
generate-certificate ca/ident1.req ca/ident1.cer
```

To import the certificate, use the following command:

```
6OS{import cert ident1 allthewinds
```

Step 5 Pre-shared keys and certificates

The identity used for the VPN has to be defined using the following command:

```
6OS{myconfig-sec}ike_id ident1
```

Assuming the 6WINDGate belongs to the `allthewinds` community, it has to be defined using the following command:

```
6OS{myconfig-sec}trust allthewinds
```

Step 6 and 7 VPN Definition and CA trusting

The following commands define a first VPN called `cer1_vpn4` for IPv4 traffic and a second one called `cer1_vpn6` for IPv6 traffic.

```
6OS{myconfig-sec}vpn cer1_vpn4 cer_strong 20.0.0.1 30.0.0.1 allthewinds
6OS{myconfig-sec}vpn cer1_vpn6 cer_strong 3abc::1 3def::1 allthewinds
```

The template used is `cer_strong`.

The IPv4 and IPv6 addresses of the VPN end-points are defined (P1 is the local gateway and P2 is the remote one).

`caname` is set to the name of the Certification Authority `allthewinds`. Associated with the identity name, it automatically selects the certificate to be used.

Use the `trust` command to specify that certificates delivered by this CA should be trusted.

```
6OS{myconfig-sec}trust allthewinds
```

Step 8 and 9 IPsec rules Definition

The IPsec rules to define are the same as for the static and IKE with pre-shared key examples.

ipsec-rule 1: A1 ⇔ B1 with AH mandatory for IPv4

```
6OS{myconfig-sec}ipsec-rule rule1 20.10.10.0/24 30.10.10.0/24 any ah tunnel
static_vpn4
```

ipsec-rule 2: P1 ⇔ P2 with ESP mandatory for IPv4

```
6OS{myconfig-sec}ipsec-rule rule2 20.0.0.1 30.0.0.1 any esp transport static_vpn4
2000
```

ipsec-rule 3: P1 ⇔ P2 with ESP mandatory for IPv6

```
6OS{myconfig-sec}ipsec-rule rule3 3abc::1 3def::1 any esp transport static_vpn4
2000
```

ipsec-rule 4: A1 ⇔ B2 with no traffic allowed for IPv4

```
6OS{myconfig-sec}ipsec-rule rule4 20.10.10.0/24 30.10.20.0/24 any discard 2000
```

ipsec-rule 5: A1 ⇔ Internet with any traffic allowed in clear for IPv4

```
6OS{myconfig-sec}ipsec-rule rule5 20.10.10.0/24 0.0.0.0/0 any discard 2100
```

ipsec-rule 6: A2 ⇔ B2 with AH and ESP mandatory for IPv6

```
6OS{myconfig-sec}ipsec-rule rule6 3abc:1:a2::/64 3def:1:b2::/64 any esp_ah tunnel
static_vpn4 2000
```

ipsec-rule 7: A1 ⇔ B2 with no traffic allowed for IPv6

```
6OS{myconfig-sec}ipsec-rule rule7 3abc:1:a1::/64 3def:1:b2::/64 any discard 2000
```

ipsec-rule 8: A1 ⇔ Internet with any traffic allowed for IPv6

```
6OS{myconfig-sec}ipsec-rule rule8 3abc:1:a1::/64 ::/0 any discard 3000
```

Step 10 Manual addition of SAs

As the VPN is not static, no manual addition of SAs is required.

Step 11 Activation of the configuration

Once the configuration has been completed, it can be applied using the `apply conf myconfig` command. The configuration is now the current configuration.

Step 12 Activation at boot time

If the current configuration behaves correctly, make it active at next boot time. The command is:

```
6OS{ }copy conf running start
```

Configuring access to roadwarrior hosts with Certificate Authentication

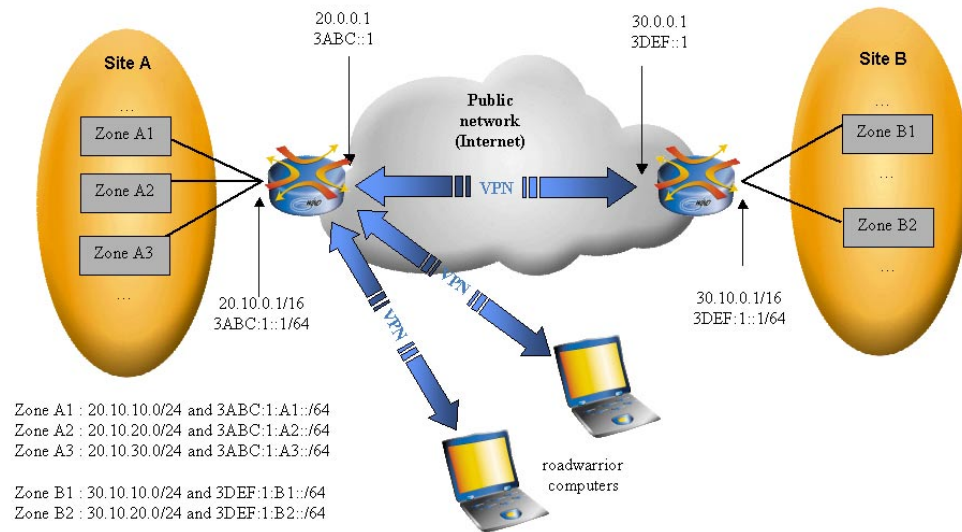


Figure 23 – Security configuration example with roadwarrior hosts

In this example, the administrator wishes to additionally accept connection of roadwarrior hosts, i.e. hosts connecting to the network from various access points, and whose IP address is unknown at configuration time. Here follows a configuration to establish L2TP/IPsec VPNs with standard PCs running Microsoft Windows XP, and authenticating themselves via certificates.

Step 1 VPN template definition

First define a vpngateway compatible with roadwarrior clients, in the root context:

```
6OS{ }vpngateway xp_cert enable main 3des sha1 "28800 sec" "6000 KB" 2 rsasig  
asn1dn 3des hmac_md5 "3600 sec" "5000 MB" none
```

Let us assume that roadwarrior hosts certificates are delivered by the same authority as other IPsec devices. The allthewinds CA and ident1 identity are already defined.

Step 2 IPsec enabling on public interface

IPsec must be enabled on the public interface:

```
6OS{myconfig-eth2_0}enable ipsec
```

Step 2 Roadwarrior VPN definition and CA trusting

Then define the roadwarrior VPN, in the sec context:

```
6OS{myconfig-sec}vpn roadw xp_cert roadwarrior allthewinds
```

Certificates delivered by the allthewinds CA should be trusted

```
6OS{myconfig-sec}trust allthewinds
```

Step 3 IPsec rules definition

Then define two IPsec rules (one for IPv4, one for IPv6) that will require that remote L2TP clients wishing to connect to the 6WINDGate first establish an IPsec connection via the roadwarrior VPN. Since this IPsec-rule is rather restrictive due to protocol and port specification, its priority index is not really significant.


```
6OS{myconfig-sec}ipsec-rule L2TP 20.0.0.1 0.0.0.0/0[1701] udp esp transport roadw  
1000  
6OS{myconfig-sec}ipsec-rule L2TP 3abc::1 ::/0[1701] udp esp transport roadw 1000
```

Provided the 6WINDGate is configured as an L2TP server, this configuration will enable remote equipments to connect to it, after negotiating a transport mode IPsec connection between the remote hosts and the 6WINDGate public addresses for the L2TP traffic. The L2TP session may carry IP traffic with private addressing.

Other security policies defined in previous paragraphs may coexist with this roadwarrior IPsec rule.

Note The roadwarrior functionality, as well as the L2TP server are advanced functions. The way they are configured may be changed without prior notice.

Chapter 25 Configuring IPv4/v6 Packet Filtering for Firewalls

This chapter describes how to configure IPv4 and IPv6 packet filtering functions in order to deploy firewall applications.

Overview

Figure 24 describes a typical firewall architecture used to protect the access to a local network. In this architecture, the 6WINDGate implements IPv4 and IPv6 filtering.

The IP filters configured in the 6WINDGate can allow, deny, forward or divert the incoming IP flows. Security can be reinforced by the deployment of application firewall proxies (HTTP, SMTP...) able to filter sessions. These proxies are located on a dedicated network in a so called DMZ zone. The servers connected on the DMZ network are the only stations that can be accessed from outside.

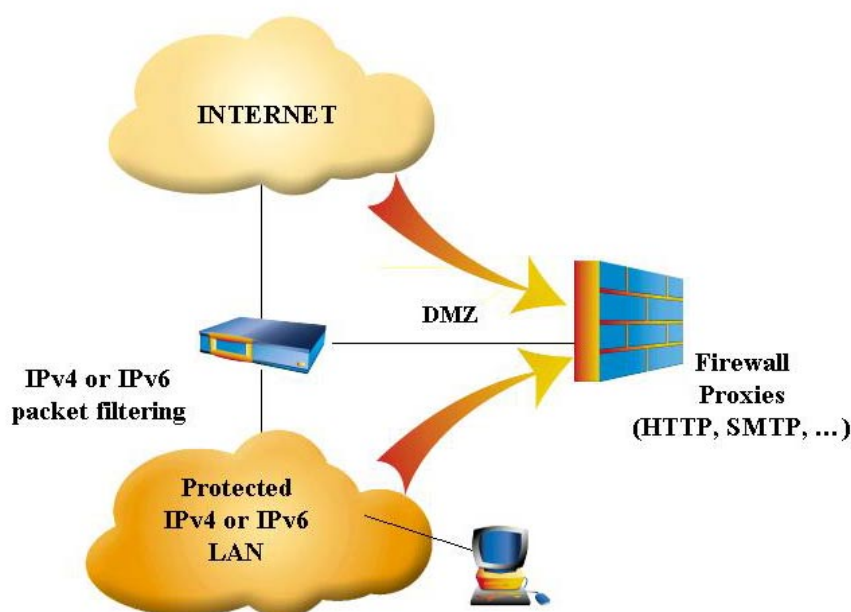


Figure 24: Firewall architecture

Packet filtering configuration consists in a sequenced list of rules. Each rule contains a packet selector and the associated action.

This enables to define a list of flows that will be enabled or disabled through the device.

There are in fact two independent list of filtering rules. One for IPv4 flows, one for IPv6 flows.

Definitions

Rules

A rule is defined by a sequence number, an action and a packet selector.

For each packet, the filter browses a rule list in the order of sequence numbers. When the first rule whose selector matches the packet is encountered, its action is applied and browsing stops. The default rule is the last one. It matches every packet and refuses them.

The possible actions are allowing, refusing, diverting and forwarding the packet.

The selector may match one or several of the following parameters: protocol (IP, TCP, UDP...), source address and mask, destination address and mask, source port list, destination port list, interface, direction (in/out), fragment flag, IP options, ICMP types, TCP states.

Dynamic Rules

The filtering module manages dynamic rules (see Rule Syntax, `keep_state` option). These rules are created with a limited lifetime and with a selector matching the parameters (addresses and ports) of the matching packet.

Dynamic rules are typically used to directly allow some legitimate traffic (e.g. TCP session). They are usually checked near the beginning of the rule set.

Stateful Management

This is done in association with the dynamic rules. The lifetime of a dynamic rule is refreshed when a matching packet is encountered.

For TCP sessions, the lifetime of the corresponding dynamic rules is also adapted to the state of the session. It's kept short when during the synchronization step and final exchange, and longer for established step.

Filter Context

All the commands related to the IPv4/v6 packet filtering have to be entered in the `fil` context.

```
6OS{myconfig}fil
```

Displaying Filter Configuration

To display information about IPv4/v6 filtering, the following commands can be used:

```
6OS{myconfig-fil}display
```

```
6OS{myconfig}display fil
```

Showing Filtering rules

A **show** command displays information reflecting current status of the 6WINDGate. This information is retrieved from the kernel unlike **display** commands, which only display configuration information.

The **show filter** command dumps IPv4 and IPv6 filtering rules currently defined in the 6WINDGate. These rules comprise manually configured rules, rules automatically generated from service flow definitions, and rules dynamically created due to session tracking (stateful filtering).

Example:

```
6OS{myconfig}show filter
=====
Packet Filtering rules for IPv4
00010 allow ip from any to any via LOOPBACK
00020 deny ip from any to 127.0.0.0/8
01000 allow udp from 10.18.18.1/24 22 to 10.18.18.11/32 550
01001 allow udp from 10.18.18.11/32 550 to 10.18.18.1/24 22
05001 allow ip from 10.16.0.113 8080 to any 8080 via eth1_0 out
65535 deny ip from any to any
=====
Packet Filtering rules for IPv6
00010 allow ipv6 from any to any via LOOPBACK
05001 allow ipv6 from 3ffe::113 8080 to any 8080 via eth1_0 out
65535 deny ipv6 from any to any
6OS{myconfig}
```

Rule Syntax

The syntax of the rules is the following:

```
6OS{myconfig-fil}{rule|rule6} [number] action_descriptor packet_selector
```

number must be within the range 10000-59999 or 1000 - 1999. Other values are reserved for automatic rules.



Note If *number* is omitted, it is automatically set to the lowest value available within the range 10000-59999. *Number* is mandatory in delta mode and for 1000-1999 range.

Action Descriptor Syntax

The *action_descriptor* parameter can be set to one of the following values:

- **allow**: Allow packets that match the rule.
- **deny**: Discard packets that match the rule.
- **unreach** *code*: Discard packets that match the rule and try to send an ICMP (or ICMP6) Destination Unreachable error message with ICMP code *code*. Known codes for IPv4 are from 0 to 16. Known codes for IPv6 are from 0 to 4.
- **reset**: TCP packets only. Discard packets that match the rule and try to send a TCP reset notice.
- **divert** *port*: Divert packets that match the rule to the divert socket bound to port *port*.

- **fwd *ipaddr* [, *port*]**: Forward the packets that match this rule to the address *ipaddr*. If the forwarding address is not one of the 6WINDGate's, the rule will only match outgoing packets.
- **check-state**: Check the packet against the dynamic rules. If a match is found, the search terminates, otherwise the filter moves to the next rule.
- **skipto *number***: Skip all following rules until the first numbered *number* or higher.

The **log** [**logamount** *log_nb*] action can be added to any of the previous actions. The effect is to display a message every time the corresponding rule is matched. The number of messages is limited to *log_nb* or 100 by default. The message contains the rule number, applied action, protocol of the packet, source address and port, destination address and port.

Packet Selector Syntax

The global syntax for the **packet_selector** parameter is the following:

```
proto from src to dst [interf_spec] [options]
```

where:

- **proto**: any IPv4 or IPv6 protocol, specified by number or by name (e.g: tcp, udp, ip, ipv6, icmp, ipv6-icmp, igmp, ...). The "all" name is used to match any protocol.
- **src** and **dst**: source and destination addresses <*ipaddress/mask*> [*ports*]
 - o For **IPv4**, the form <*ipaddress/mask*> may be of the following:
 - IP address like 1.2.3.4 . Only exact addresses will match the rule
 - IP address with mask: 1.2.3.4/24 . Addresses from 1.2.3.0 to 1.2.3.255 will match the rule.
 - IP address with mask: 1.2.3.4:255.255.240.0. Addresses from 1.2.0.0 to 1.2.15.255 will match the rule
 - o For **IPv6**, the form <*ipaddress/prefix*> may be of the following:
 - IPv6 address 1234:5678::1111. Only exact addresses will match the rule
 - IPv6 address with mask: 1234::5678:2222/112 . Addresses from 1234::5678:0000 to 1234::5678:ffff will match the rule.
 - o In both cases (IPv4/IPv6), the **not** modifier can invert the match. That does not affect the selection of port numbers:
- **ports**: For TCP or UDP packets only. This can be one single port specified by a number or a service name (isakmp, ripng, ...), a range of port (500-724) or a list of the preceding (500-600, 22,8080).
- **interf_spec**: Some combination of the following:
 - o **in**: Only match incoming packets.
 - o **out**: Only match outgoing packets.
 - o **via *ifx***: Packet must go through interface *ifx*.
 - o **via any**: Packet must go through some interface.

The **packet_selector** parameter options are the following:

- **keep-state**: If the packet matches, this option will create a dynamic rule that will allow matching bidirectional traffic between *src* and *dst* (including *ports*) using the same protocol. The dynamic rule has a limited lifetime which is refreshed every time a matching packet is found.
- **frag**: The rule matches if the packet is a fragment (but not the first) of a datagram. This option is not compatible with port specification.

- **ipoptions** *list_opt*: IPv4 packets only. The packet matches if the IP header contains the comma separated list of options specified in *list_opt*. The supported options are: **ssrr** (strict source route), **lsrr** (loose source route), **rr** (record packet route), and **ts** (timestamp). The absence of a particular option may be denoted with a '!'.
- **ip6options** *list_opt*: IPv6 packets only. The packet matches if the IP header contains the comma separated list of options specified in *list_opt*. The supported options are: **hopopt** (Hop-by_hop header), **route** (Routing header), **frag** (Fragment header), **esp** (Encapsulating Security Payload header), **ah** (Authentication header), **nonxt** (No next header) and **opts** (Destination options header). The absence of a particular option may be denoted with a '!'.
- **established**: TCP packets only. Matches packets that have the RST or ACK bits set.
- **setup**: TCP packets only. Matches packets that have the SYN bit set but no ACK bit.
- **tcpflag** *list_flg*: TCP packets only. The packet matches if the TCP header contains the comma separated list of flags specified in *list_flg*. The supported flags are: **fin**, **syn**, **rst**, **psh**, **ack** and **urg**. The absence of a particular flag may be denoted with a '!'.
- **icmpatypes** *list_type*: ICMP packets only. The packet matches if the ICMP type is in the comma separated list of types specified in *list_type*. The supported types are: echo reply (0), destination unreachable (3), source quench (4), redirect (5), echo request (8), router advertisement (9), router solicitation (10), time-to-live exceeded (11), IP header bad (12), timestamp request (13), timestamp reply (14), information request (15), information reply (16), address mask request (17) and address mask reply (18).
- **icmp6types** *list_type*: ICMP6 packets only. The packet matches if the ICMP6 type is in the comma separated list of types specified in *list_type*. The supported types are: Destination unreachable (1), Packet too big (2), Time exceeded (3), Parameter Problem (4), Echo request (128), Echo reply (129), Group membership query (130), Group membership report (131), Group membership reduction (132), Router solicit (133), Router advert (134), Neighbour solicit (135), Neighbour advert (136), Redirect (137), Router renumbering (138).

Packet Filtering and Service Flow Configuration

The service flow context (**sfl**) alleviates the configuration of filtering rules relating to control and management traffic. It provides commands that will automatically generate the required filtering rules to allow control traffic.

These commands are of the form `flow_security {allow|protected}`.

One or more passing rules will automatically be generated for flows whose security parameter is set to **allow**. These rules appear when the `show filter` command is invoked.

No automatic rule will be generated for flows whose security parameter is set to **protected**. The user may choose to manually add an explicit rule for this flow in the **sfl** context.

Example:

```
6OS{myconfig-sfl}dns_security allow
```

The following command allows IKE traffic (Internet Key Exchange) when it is set to **clear**:

```
6OS{myconfig-sfl}ike_security {clear|protected}
```

The following command allows SSH traffic (Secure SHell) when it is set to `clear`:

```
60S{myconfig-sf1}ssh_security {clear|protected}
```

The following command allows DNS traffic (Domain Name Service) when it is set to `allow`:

```
60S{myconfig-sf1}dns_security {allow|protected}
```

The following command allows ICMP error packets when it is set to `allow`:

```
60S{myconfig-sf1}icmp_errors_security {allow|protected}
```

The following command allows the 6WINDGate to ping remote hosts when it is set to `allow`:

```
60S{myconfig-sf1}icmp_echo_out_security {allow|protected}
```

The following command allows remote hosts to ping the 6WINDGate when it is set to `allow`:

```
60S{myconfig-sf1}icmp_echo_in_security {allow|protected}
```

The following command allows ICMPv6 error packets when it is set to `allow`:

```
60S{myconfig-sf1}icmpv6_errors_security {allow|protected}
```

The following command allows the 6WINDGate to ping6 remote hosts when it is set to `allow`:

```
60S{myconfig-sf1}icmpv6_echo_out_security {allow|protected}
```

The following command allows remote hosts to ping6 the 6WINDGate when it is set to `allow`:

```
60S{myconfig-sf1}icmpv6_echo_in_security {allow|protected}
```

The following commands add or remove multicast groups for which IGMP traffic is allowed. An `igmp_group` command enables joins, leaves and membership summaries for all multicast groups matching the prefix.

```
60S{myconfig-sf1}igmp_group addr/mask
60S{myconfig-sf1}delete igmp_group addr/mask
```

The following commands add or remove IPv6 multicast groups for which MLD traffic is allowed. An `mld_group` command enables joins, leaves and membership summaries for all multicast groups matching the prefix.

```
60S{myconfig-sf1}mld_group addr/mask
60S{myconfig-sf1}delete mld_group addr/mask
```

Packet Filtering and Migration Configuration

The packet filter does not apply to encapsulating packet for migration tunnels.

It means that only the encapsulated packets are going through the filter (the encapsulating IPv4 or IPv6 headers are removed before entering the filter).

So the rules must be written without taking into account the migration tunnels.

Log Messages for IP Filtering

The log messages for IP filtering are configured in the `fil` context

The following commands make enable to send and to stop IP filtering log messages to a log session. There is no severity parameter in this case.

```
6OS{myconfig-fil}log filter sessionX
6OS{myconfig-fil}delete log filter sessionX
```

You need to have a log session *sessionX* already created, please refer to the “Configure the logging service” paragraph.

Example:

```
6OS{myconfig-fil}log filter session1
6OS{myconfig-fil}delete log filter session1
```

The content of the *log* file (*session1* in our example) is the dump of the packets matching the filtering rules with *log* action.

Usually, one chooses to log the “deny” rules to keep a trace of illegal packets received and detect possible attacks.

Use the *log* keyword of *rule* and *rule6* commands produce messages when packets match an IP filter rule.

Example:

```
6OS{myconfig-fil}rule 30000 deny log udp from any to any
```

or

```
6OS{myconfig-fil}rule 59000 deny log logamount 30 all from any to any
```



Caution A huge amount of log messages may decrease the performance of the 6WINDGate.

Chapter 26 Configuring a PPP Server

This chapter describes how to configure a PPP Server on a 6WINDGate.

Overview

The PPP server opens multiple PPP sessions dynamically, on demand. During the opening of the PPP session, the server provides to the client for several items such as IPv4 and IPv6 addresses, DNS addresses and MRU.

Before that, an authentication phase may occur, during which the client and/or the server will authenticate themselves.

The access and authentication data can be locally defined, or provided by an AAA RADIUS server (Authentication, Authorization and Accounting).

For each PPP session, a Point-to-point interface (`pppi`) is automatically created. The routes associated to that interface may be automatically handled by the 6WINDGate.

If the PPP server runs in **unnumbered** mode, the PPP interfaces are set without addresses. In this case, the routes to the clients are automatically set by the 6WINDGate.

If the PPP server address is needed on some interface, a loopback interface must be configured with this address.

The PPP traffic is encapsulated over a level 2 transport layer, which typically is L2TP (Layer Two Transport Protocol).

Configuration Contexts

The complete configuration of a PPP server involves several contexts : The `ppp_serveri` context for the PPP server parameters, the `l2tpi` context for the transport layer configuration, the `aaa` context for the local AAA database.

There can be several PPP servers (`ppp_server0`, `ppp_server1`, ...). For each, there must be one distinct L2TP server (`l2tp0`, `l2tp1`, ...)

The `aaa` context is unique and is shared by all the PPP servers.

Configuring the PPP Server

Setting the PPP server up and down

A PPP server can be set up and down using the following command:

```
6OS{myconfig-ppp_server0}service {up|down}
```

By default the server is **down**.

Setting the transport layer

To configure the transport layer, use the following command:

```
6OS{myconfig-ppp_serveri}over l2tpj
```



Note

There is no straightforward relation between the index of the `ppp_server` and the one of the L2TP server. For example, it is possible in the `ppp_server1` context to provide PPP sessions over `l2tp0`.

Setting the PPP server address

The following command sets the PPP server address:

```
6OS{myconfig-ppp_server0}ipaddress ipaddress
```

This address is one of two endpoints of the Point-to-point interfaces (`pppi`)

Enabling IPv4 PPP sessions

To enable IPv4 PPP sessions, use the following command:

```
6OS{myconfig-ppp_server0}enable ipv4
```

Enabling IPv6 PPP sessions

To enable IPv6 PPP sessions, use the following command:

```
6OS{myconfig-ppp_server0}enable ipv6
```

Setting PPP interface MTU

To set the MTU of the PPP interface, use the following command:

```
6OS{myconfig-ppp_server0}mtu {default|value}
```

Setting PPP interface MRU

To set the MRU of the PPP interface, use the following command:

```
6OS{myconfig-ppp_server0}mru {default|value}
```

Setting PPP interface TCPMSS

To set the TCPMSS (v4 and v6) of the PPP interface, use the following command:

```
6OS{myconfig-ppp_server0}tcp4mss {disable|automatic|value}
```

```
6OS{myconfig-ppp_server0}tcp6mss {disable|automatic|value}
```

For most common configurations, the default values for `mtu`, `mru`, `tcp4mss` and `tcp6mss` are convenient.


For more details, please refer to ADSL-PPPoE application note.

Setting the pool of addresses

For each PPP session, an IPv4 address is attributed to the client. This address is taken from a pool.

To define the pool of addresses, use the following command:

```
6OS{myconfig-ppp_server0}ippool ipaddress_low ipaddress_high
6OS{myconfig-ppp_server0}ippool ipaddress/mask
```


 **Note** If IPv6 PPP sessions are enabled, the IPv6 address is automatically generated, and can be chosen by the client.

Setting the DNS parameters

To define the DNS addresses, use the following commands:

```
6OS{myconfig-ppp_server0}primary_dns ipaddress
6OS{myconfig-ppp_server0}secondary_dns ipaddress
```

Those addresses are sent to the PPP client.

 **Note** The DNS addresses sent to the PPP client may be different from the DNS address used by the 6WINDGate (**nameserver** command).

Configuring PPP authentication parameters

The PPP server provides several levels of authentication : PAP, CHAP, client and/or server authentication.

To set the PAP or CHAP authentication for the client, use the following commands:

```
6OS{myconfig-ppp_server0}pap_ingress {required|none}
6OS{myconfig-ppp_server0}chap_ingress {required|none}
```

To set the PAP or CHAP authentication for the server, use the following commands:

```
6OS{myconfig-ppp_server0}pap_egress {accepted|none}
6OS{myconfig-ppp_server0}chap_egress {accepted|none}
6OS{myconfig-ppp_server0}serverid server_login server_password
```

Of course, the authentication parameters of the client (login, password) must be known by the server. These parameters can be locally stored (see **aaa** context) or provided by a RADIUS server.

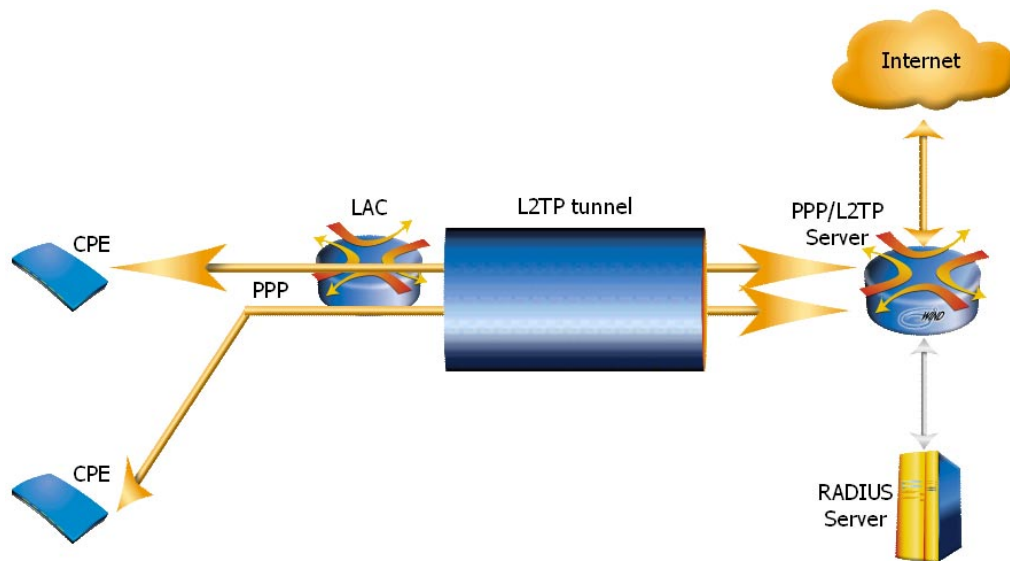
Configuring A L2TP Server

Overview

L2TP provides a level 2 transport layer between distant hosts, thus enabling the establishment of PPP sessions between routers that are not directly connected.

The L2TP layer is generally set between an L2TP Network Server (LNS) and an L2TP access concentrator (LAC).

In this scheme, the PPP client originates a PPP session over Ethernet or ATM (for example) to the LAC. The LAC mounts an L2TP connection to the corresponding PPP server, then forwards the PPP frames directly to the PPP server, through the L2TP tunnel.



For each LNS-LAC couple, a unique L2TP tunnel is set.

Inside an L2TP tunnel, several L2TP sessions can appear, one for each PPP session.

The L2TP client and server can authenticate themselves during the opening of the session if a secret is shared.

The control messages of the L2TP session can also be encrypted, using the same secret.

The data frames of the L2TP session (PPP packets) can be sequenced, i.e. packets arriving out of order are dropped.

Note The authentication at L2TP level is independent (and different) from the authentication mechanisms at PPP level. They are performed between different routers.

Configuration

The L2TP server is automatically activated when a “`over l2tp i`” command is encountered for a PPP server.

To configure the IPv4 address of the L2TP server, use the following command:

```
6OS{myconfig-l2tp0} ipaddress ipv4address
```

To set the encryption mode on the L2TP server, use the following command :

```
6OS{myconfig-l2tp0} enable hiding
```

To set the sequencing mode, use the following command :

```
6OS{myconfig-l2tp0} enable sequencing
```

To set a shared secret for authentication and encryption, use the following command :

```
6OS{myconfig-l2tp0} shared_secret client_ipv4address thesecret
```

Example 1:

```
6OS{myconfig} ppp_server0  
6OS{myconfig-ppp_server0} service up
```

```

6OS{myconfig-ppp_server0}over l2tp0
6OS{myconfig-ppp_server0}enable ipv4
6OS{myconfig-ppp_server0}enable ipv6
6OS{myconfig-ppp_server0}disable unnumbered
6OS{myconfig-ppp_server0}mtu default
6OS{myconfig-ppp_server0}mru default
6OS{myconfig-ppp_server0}tcp4mss disable
6OS{myconfig-ppp_server0}tcp6mss disable
6OS{myconfig-ppp_server0}ipaddress 64.235.12.1
6OS{myconfig-ppp_server0}primary_dns 112.35.56.4
6OS{myconfig-ppp_server0}secondary_dns 112.35.56.8
6OS{myconfig-ppp_server0}pap_ingress required
6OS{myconfig-ppp_server0}pap_egress accepted
6OS{myconfig-ppp_server0}serverid myppplogin myppppassword
6OS{myconfig-ppp_server0}aaa_radius main1 64.235.20.1 both myradiuspassword 2 3 1
6OS{myconfig-ppp_server0}aaa_radius main2 64.235.20.2 both myradiuspassword 2 3 2
6OS{myconfig-ppp_server0}aaa_radius_optional opt1 64.235.20.3 myradiuspassword 2
3 1
6OS{myconfig-l2tp0}ipaddress 164.35.120.1
6OS{myconfig-l2tp0}disable hiding
6OS{myconfig-l2tp0}disable sequencing
6OS{myconfig-l2tp0}shared_secret 83.65.4.125 secretofLAC1
6OS{myconfig-l2tp0}shared_secret 43.56.201.12 secretofLAC2

```

Example 2:

```

6OS{myconfig}ppp_server0
6OS{myconfig-ppp_server0}service up
6OS{myconfig-ppp_server0}over l2tp0
6OS{myconfig-ppp_server0}enable ipv4
6OS{myconfig-ppp_server0}enable ipv6
6OS{myconfig-ppp_server0}enable unnumbered
6OS{myconfig-ppp_server0}mtu default
6OS{myconfig-ppp_server0}mru default
6OS{myconfig-ppp_server0}tcp4mss disable
6OS{myconfig-ppp_server0}tcp6mss disable
6OS{myconfig-ppp_server0}ipaddress 64.235.12.1
6OS{myconfig-ppp_server0}ippool 64.235.12.10 64.235.13.254
6OS{myconfig-ppp_server0}ippool 64.235.14.0/24
6OS{myconfig-ppp_server0}primary_dns 112.35.56.4
6OS{myconfig-ppp_server0}secondary_dns 112.35.56.8
6OS{myconfig-ppp_server0}chap_ingress required
6OS{myconfig-ppp_server0}chap_egress none
6OS{myconfig-ppp_server0}serverid myppplogin myppppassword

6OS{myconfig-l2tp0}ipaddress 164.35.120.1
6OS{myconfig-l2tp0}disable hiding
6OS{myconfig-l2tp0}disable sequencing
6OS{myconfig-l2tp0}shared_secret 83.65.4.125 secretofLAC1
6OS{myconfig-l2tp0}shared_secret 43.56.201.12 secretofLAC2

6OS{myconfig-aaa}user_id login1 password1 64.235.14.23/24
6OS{myconfig-loopback}loop 1 64.235.12.1/24

```

Chapter 27 Configuring a AAA local database


Configuring a AAA (Authentication Authorization Accounting) local database enables to manage network parameters of a few client hosts.

The authentication local database

The **aaa** context describes the local database for authentication parameters. Each entry of the database contains the login of a client, its password, an optional IPv4 address and optional IPv4 and/or IPv6 routes.

To set the authentication parameters for a client, use the following command:


```
6OS{myconfig-aaa}userid client_login client_password
```

 **Note** If a radius server is configured, the local database is ignored.

Configuring a static address for a client

To specify a static IPv4 address to a client, use the following command:

```
6OS{myconfig-aaa}userid client_login client_password ipv4address
```

 **Note** It is necessary to configure PAP or CHAP authentication to specify a static address. When a static address is configured, the `ippool` parameter is ignored.

Configuring IPv4 and IPv6 routes for a client

To specify an IPv4 route to a client, use the following command:

```
6OS{myconfig-aaa}userid client_login client_password ipv4address/mask
```

In this case, the route to `ipv4address/mask` is automatically set when the client has been authorized.

 **Note** It is necessary to configure PAP or CHAP authentication to specify a route.

To specify an IPv6 route to a client, use the following command:

```
6OS{myconfig-aaa}userid client_login client_password ipv6prefix/mask
```

In this case, the route to `ipv6prefix/mask` is automatically set when the client has been authorized.

Chapter 28 Configuring the RADIUS client

Configuring the access to a remote RADIUS server enables to request network parameters of potentially numerous client hosts, from a RADIUS server.

Access to RADIUS servers

A RADIUS server provides the same information as the local database and more.

When a RADIUS server is configured, the local pool of addresses and the local AAA database is ignored and provided by the RADIUS server.

The RADIUS server provides the IPv4 address for the client and the IPv6 Prefix for building the clients IPv6 address.

Two types of RADIUS servers are available.

- The “main” type can be used for accounting and/or authentication.
- The “optional” type only serves for access and authentication. It is typically used for an IPv6 database.

Several RADIUS servers can be configured. They are sorted by priority order.

To configure a main RADIUS server, use the following command:

```
6OS{myconfig-ppp_server0}aaa_radius name IPv4address {accounting/auth/both}  
secret timeout nb_retry priority
```

To configure an optional RADIUS server, use the following command:

```
6OS{myconfig-ppp_server0}aaa_radius_optional name IPv4address secret timeout  
nb_retry priority
```

**Note**

It is important to configure a timeout coherent with the internal timeout of PPP protocol. We recommend a value lower than 3 seconds.

Chapter 29 Troubleshooting the 6WINDGate – Displaying current status

Overview

Most of the 6WINDGate functions provide a status display command. Status display commands names all start with a `show` keyword. They are called from the root context. They differ from `display` commands: `display` commands recapitulate the configuration (static information), `show` commands present the current status, dynamic values or objects (dynamic information).

Main network services

Displaying running interfaces

The `show interface` command displays running network interfaces. Interfaces may be physical interfaces (e.g. `eth0_0`) or pseudo-interfaces (e.g. `ppp0`, `stu0`, `bnet0`, `tu0...`).

```
6OS{ }show interface [ifacename [stats [detail]]]
```

Information can also be requested about a specific interface by specifying its name. The interface may not be running.

To display traffic statistics of an interface, add the `stats` keyword.

More detailed statistics (about lower layers) may be requested by adding the `detail` keyword. All interfaces do not provide detailed statistics.

Example:

```
6OS{ }show interface
eth0_0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
  ether 00:90:fb:04:07:1b
  media: Ethernet autoselect (10baseT/UTP <full-duplex>)
eth2_0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
  ether 52:54:40:20:6d:f0
eth1_0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
  inet 10.16.0.233 netmask 0xfffffff0 broadcast 10.16.0.255
  ether 00:50:04:36:8c:79
```

Automatic tunnel:

```
tu0: flags=1041<UP,RUNNING,LINK0> mtu 1480
  inet6 ::127.0.0.1 /96
  lladdr 7f:00:00:01
wild4: flags=8041<UP,RUNNING,MULTICAST> mtu 1480
```

```
wild6: flags=8041<UP,RUNNING,MULTICAST> mtu 1480
bnet0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.11.12.13 netmask 0xffffffff broadcast 10.11.12.255
    ether 02:09:c0:40:6b:88

6OS{ }show interface eth0_0
eth0_0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:90:fb:04:07:86
    media: Ethernet autoselect (10baseT/UTP <full-duplex>)

6OS{ }show interface eth0_0 stats
stats:
    1157 inbound packets
    0 inbound errors
    89480 inbound bytes
    6 outbound packets
    0 outbound errors
    260 outbound bytes
    0 collisions
    0 dropped packets

6OS{ }show interface bnet0 stats detail
stats:
    0 inbound packets
    0 inbound errors
    0 outbound packets
    5 outbound errors
    0 outbound bytes
    420 collisions
    0 dropped packets
detailed stats:
    bridge bridge_0
        link0 to eiface eiface_0
            42 recvOctets
            1 recvPackets
            1 recvBroadcast
        link1 to vlan vlans_r10
            42 xmitOctets
            1 xmitPackets
            1 xmitBroadcasts
        link2 to vlan vlans_ep0
            42 xmitOctets
            1 xmitPackets
            1 xmitBroadcasts
    vlan vlans_ep0
        46 xmitOctets
        1 xmitPackets
    etf emux_ep0
        1 packets_out
    vlan vlans_r10
        46 xmitOctets
        1 xmitPackets
    etf emux_r10
        1 packets_out
```

Displaying running services

The `show service` command displays active and inactive services.

```
6OS{}show service
Service SSH is active
Service SNMP is inactive
Service NAT is inactive
Service NATPT is inactive
Service FILTER is inactive
Service RIP is active
Service RIPng is active
Service OSPFv2 is inactive
Service OSPFv3 is inactive
Service BGP is inactive
Service TELNET is inactive
Service IP Forwarding is active
Service IPv6 Forwarding is active
Service IPSEC is inactive
Service DHCPSEVER is inactive
Service DHCP is inactive
Service NTP is inactive
Service Mobility home agent is active
```

Displaying and flushing the ARP and NDP caches

The `show service` command also display the content of the ARP and NDP caches. The database contains both statically configured ARP and NDP entries and entries learnt by the ARP and NDP protocols.

```
6OS{}show service
```

Example:

```
6OS{}show service
...
ARP Table:
hurricane.6wind.com (192.168.0.1) at 0:20:40:54:12:d0 [ethernet]
down.6wind.com (192.168.0.12) at (incomplete) [ethernet]
? (192.168.0.20) at 0:2:b3:71:28:7 [ethernet]
? (192.168.0.47) at 0:9:c0:40:c0:2d [ethernet]

NDP Table:
          Address IPv6                      MAC                      Interf
fe80::240:33ff:fe54:b1a0          00:40:33:54:B1:A0          eth2_0
```

The `flush arp` and `flush ndp` commands respectively remove all the ARP and NDP dynamic entries. The statically configured ARP and NDP entries remain in the cache.

```
6OS{}flush arp
6OS{}flush ndp
```

Displaying Network connections

The `show system connections` command displays running IP network connections.

```
6OS{}show system connections
```

Example:

```
6OS{}show system connections
Active Internet connections
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	(state)
tcp	0	0	10.22.3.103.ssh	10.22.3.51.1146	ESTABLISHED
tcp	0	0	127.0.0.1.61225	*.*	LISTEN
udp	0	0	10.22.3.103.isakmp	*.*	
udp	0	0	10.23.3.103.isakmp	*.*	
udp	0	0	10.16.0.88.isakmp	*.*	
udp	0	0	127.0.0.1.isakmp	*.*	

Displaying Network statistics

The `show system statistics` command displays protocol-related statistics. An optional protocol name may be specified to restrict statistics to the provided protocol. *protocol* is one of `ipv4`, `icmpv4`, `igmp`, `udpv4`, `tcpv4`, `ipv6`, `icmpv6`, `udpv6`, `tcpv6`

```
6OS{}show system statistics [protocol]
```

Example:

```
6OS{}show system statistics ipv4
```

```
ip:
```

```
10132 total packets received
0 bad header checksums
0 with size smaller than minimum
0 with data size < data length
0 with header length < data size
0 with data length < header length
0 with bad options
0 with incorrect version number
0 fragments received
0 fragments dropped (dup or out of space)
0 fragments dropped after timeout
0 packets reassembled ok
5106 packets for this host
2486 packets for unknown/unsupported protocol
2186 packets forwarded (0 packets fast forwarded)
267 packets not forwardable
29 packets received for unknown multicast group
1457 redirects sent
2398 packets sent from this host
0 packets sent with fabricated ip header
0 output packets dropped due to no bufs, etc.
3 output packets discarded due to no route
0 output datagrams fragmented
0 fragments created
0 datagrams that can't be fragmented
```

```
6OS{}show system statistics icmpv4
```

```
icmp:
```

```
0 packet too big sent
0 time exceeded sent
0 parameter problems sent
0 redirects sent
2 echo requests sent
8 echo replies sent
0 group queries sent
3 group reports sent
0 group terminations sent
0 router solicitations sent
1 router advertisements sent
3 neighbor solicitations sent
```

```

2 neighbor advertisements sent

0 icmpv6_code out of range sent
0 Packet < ICMPV6_MINLEN
0 Bad checksum
0 Calculated bound mismatch
0 Number of responses
0 unreachable received
0 packet too big received
0 time exceeded received
0 parameter problems received
0 redirects received
8 echo requests received
2 echo replies received
0 group queries received
0 group reports received
0 group terminations received
0 bad group queries received
0 bad group reports received
0 our groups' reports received
0 bad group terminations received
0 router solicitations received
1 router advertisements received
2 neighbor solicitations received
2 neighbor advertisements received
0 bad router solicitations received
0 bad router advertisements received
0 bad neighbor solicitations received
0 bad neighbor advertisements received
0 bad neighbor advertisements received
    
```

Memory

Displaying system virtual memory

The `show system virtual memory` command displays statistics about the usage of virtual memory on the 6WINDGate.

Example:

```

6OS{ }show system virtual memory
Memory statistics by bucket size
Size      In Use   Free    Requests  HighWater  Couldfree
 16      10815   193     114136    1280       0
 32        328    56     1092180    640       0
 64       1144   136     88531     320       0
128        471    73     10436     160       0
256        907   149     11115      80       8
512        113    23       675       40      14
1K         128    32    1080335     20       0
2K         115     1     90402      10       0
4K          11     1    3238904      5       0
8K          22     1        36       5       0
16K         9      0         17       5       0
64K         1      0          1       5       0
    
```

512K 1 0 3 5 0

Memory usage type by bucket size

```
Size Type(s)
16 key mgmt, netgraph, kld, devbuf, temp, proc, sysctl, soname, pcb,
    vnode, ifaddr, ether_multi, routetbl, IpFw/IpAcct, IpFw/IpAcct
32 key mgmt, netgraph, kld, sigio, devbuf, temp, pgrp, subproc, sysctl,
    soname, pcb, vnode, ifaddr, ether_multi, routetbl, in_multi
64 netgraph, file, lockf, namecache, devbuf, temp, session, pcb, vnode,
    ifaddr, ether_multi, routetbl, MFS node, maddr
128 key mgmt, netgraph, kld, timecounter, file desc, zombie, namecache,
    devbuf, temp, proc, cred, ttys, soname, vnode, ifaddr, routetbl,
    VM pgdata, ZONE
256 netgraph, file desc, devbuf, temp, subproc, iocltlops, pcb, vnode,
    ifaddr, routetbl, IpFw/IpAcct, NFS srvsoc, NFS daemon, FFS node,
    MFS node, VM pgdata, IpFw/IpAcct
512 key mgmt, netgraph, file desc, devbuf, temp, proc, iocltlops,
    BIO buffer, mount, ifaddr, UFS mount, VM pgdata
1K key mgmt, kld, file desc, temp, iocltlops, BIO buffer, NQNF Lease
2K kld, file desc, devbuf, temp, ttys, pcb, BIO buffer, ifaddr,
    UFS mount
4K netgraph, kld, devbuf, temp, iocltlops
8K temp, NFS hash, UFS ihash
16K kld, namecache, devbuf, temp, mbuf
64K kld
512K kld, temp
```

Memory statistics by type

Type	InUse	MemUse	HighUse	Limit	Requests	Type Limit	Kern Limit	Size(s)
key mgmt	10039	171K	180K	6144K	10606	0	0	16,32,128,512,1K
netgraph	33	3K	7K	6144K	237	0	0	16,32,64,128,256,512,4K
kld	32	490K	495K	6144K	95	0	0	16,32,128,1K,2K,4K,16K,64K,512K
timecounter	10	2K	2K	6144K	10	0	0	128
file desc	25	9K	9K	6144K	3799	0	0	128,256,512,1K,2K
file	82	6K	6K	6144K	57338	0	0	64
sigio	1	1K	1K	6144K	1	0	0	32
zombie	0	0K	1K	6144K	3663	0	0	128
lockf	1	1K	1K	6144K	3	0	0	64
namecache	798	66K	75K	6144K	14782	0	0	64,128,16K
devbuf	204	117K	117K	6144K	225	0	0	16,32,64,128,256,512,2K,4K,16K
temp	1041	251K	982K	6144K	5608015	0	0	16,32,64,128,256,512,1K,2K,4K,8K,16K,512K
pgrp	19	1K	1K	6144K	484	0	0	32
session	18	2K	2K	6144K	397	0	0	64
proc	5	2K	2K	6144K	6	0	0	16,128,512
subproc	58	4K	5K	6144K	7701	0	0	32,256
cred	11	2K	2K	6144K	1364	0	0	128
sysctl	0	0K	1K	6144K	245	0	0	16,32
iocltlops	0	0K	4K	6144K	16	0	0	256,512,1K,4K
ttys	255	36K	41K	6144K	915	0	0	128,2K
soname	15	1K	1K	6144K	4435	0	0	16,32,128
pcb	26	7K	7K	6144K	1009	0	0	16,32,64,256,2K
BIO buffer	126	129K	160K	6144K	681	0	0	512,1K,2K
mount	3	2K	2K	6144K	3	0	0	512

vnodes	36	6K	6K	6144K	563	0	0	16,32,64,128,256
ifaddr	99	161K	161K	6144K	130	0	0	16,32,64,128,256,512,2K
ether_multi	39	2K	2K	6144K	55	0	0	16,32,64
routetbl	186	20K	21K	6144K	872	0	0	16,32,64,128,256
in_multi	4	1K	1K	6144K	4	0	0	32
IpFw/IpAcct	8	2K	2K	6144K	56	0	0	16,256
NFS srvsock	2	1K	1K	6144K	2	0	0	256
NFS daemon	1	1K	1K	6144K	1	0	0	256
NQNFS Lease	1	1K	1K	6144K	1	0	0	1K
NFS hash	1	8K	8K	6144K	1	0	0	8K
FFS node	43	11K	11K	6144K	81	0	0	256
MFS node	775	194K	230K	6144K	8841	0	0	64,256
UFS ihash	1	8K	8K	6144K	1	0	0	8K
UFS mount	6	9K	9K	6144K	6	0	0	512,2K
VM pgdata	25	4K	5K	6144K	27	0	0	128,256,512
ZONE	20	3K	3K	6144K	20	0	0	128
mbuf	1	12K	12K	6144K	1	0	0	16K
maddr	5	1K	1K	6144K	13	0	0	64
IpFw/IpAcct	10	2K	2K	6144K	66	0	0	16,256
Memory Totals:	In Use	Free	Requests					
	1731K	118K	5726771					

Show boot messages

The `show log system` command displays messages generated by the 6WINDGate during the last boot process.

Example:

```
6OS{ }show log system
Copyright (c) 1992-1999 FreeBSD Inc.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California. All rights reserved.
SIXOS 6.3.1-RELEASE #1: Mon Jul 21 16:14:42 CEST 2003
root@6wind.com:/6wind
Timecounter "i8254" frequency 1193087 Hz
Timecounter "TSC" frequency 334065744 Hz
CPU: AMD-K6(tm) 3D processor (334.07-MHz 586-class CPU)
  Origin = "AuthenticAMD" Id = 0x589 Stepping = 9
  Features=0x8021bf<FPU,VME,DE,PSE,TSC,MSR,MCE,CX8,PGE,MMX>
  AMD Features=0x80000800<SYSCALL,3DNow!>
real memory = 67108864 (65536K bytes)
avail memory = 27049984 (26416K bytes)
Bad DMI table checksum!
Preloaded elf kernel "6200-6.3.1dbg1-s0.6os" at 0xc24ca000.
Preloaded userconfig_script "/boot/kernel.conf" at 0xc24ca0ac.
Probing for devices on PCI bus 0:
chip0: <Intel 82439TX System Controller (MTXC)> rev 0x01 on pci0.0.0
chip1: <Intel 82371AB PCI to ISA bridge> rev 0x02 on pci0.7.0
ide_pci0: <Intel PIIX4 Bus-master IDE controller> rev 0x01 on pci0.7.1
chip2: <Intel 82371AB Power management controller> rev 0x02 on pci0.7.3
rl0: <RealTek 8139 10/100BaseTX> rev 0x10 int a irq 10 on pci0.9.0
rl0: Ethernet address: 00:90:fb:04:05:2c
rl0: autoneg complete, link status good (full-duplex, 10Mbps)
```

```
vga0: <Chips & Technologies model 00c0 VGA-compatible display device> rev 0x64 i
nt a irq 15 on pci0.10.0
Probing for PnP devices:
Probing for devices on the ISA bus:
sc0 on isa
sc0: VGA color <16 virtual consoles, flags=0x0>
ed0 at 0x280-0x29f irq 11 on isa
ed0: address 52:54:40:20:0e:53, type NE2000 (16 bit)
atkbdc0 at 0x60-0x6f on motherboard
atkbd0 irq 1 on isa
sio0 at 0x3f8-0x3ff irq 4 flags 0x30 on isa
sio0: type 16550A, console
wdc0 at 0x1f0-0x1f7 irq 14 on isa
wdc0: unit 0 (wd0): <QUANTUM FIREBALLlct08 08>
wd0: 8063MB (16514064 sectors), 16383 cyls, 16 heads, 63 S/T, 512 B/S
ppc0 at 0x378 irq 7 flags 0x40 on isa
ppc0: Generic chipset (NIBBLE-only) in COMPATIBLE mode
ppi0: <generic parallel i/o> on ppbus 0
1 3C5x9 board(s) on ISA found at 0x300
ep0 at 0x300-0x30f irq 9 on isa
ep0: au/utp/bnc[*UTP*] address 00:60:97:9e:50:e1
fla0 not found
npx0 on motherboard
npx0: INT 16 interface
sio1 at 0x2f8-0x2ff irq 3 on isa
sio1: type 16550A
IPv6 packet filtering initialized, divert enabled, rule-based forwarding enabled
, default to deny, logging limited to 100 packets/entry by default
IP packet filtering initialized, divert enabled, rule-based forwarding enabled,
default to deny, logging limited to 100 packets/entry by default
changing root device to wd0s1a
rootfs is 34208 Kbyte compiled in MFS
IPsec version S0 - algo=des aes
@(#) IPLUG_3_4_6_m2
```

Routing status display

To cope with the richness of routing functions, status display procedures are regrouped into a **show-route** context. One can enter this context by typing **show route** in the root context. The available **show** commands and arguments may be listed by using the **?** character

Example:

```
6OS{show route
6OS{show-route}?
    exit Exit the show routing context
    show Show running system information
6OS{show-route}show ?
    bgp      BGP information
    cpu      Thread CPU usage
    debugging Debugging functions (see also 'undebug')
    interface Interface status and configuration
    ip       IP information
    ipv6     IPv6 information
    memory   Memory statistics
6OS{show-route}show ip ?
```



```

access-list      List IP access lists
bgp              BGP information
community-list  List community-list
extcommunity-list List extended-community list
forwarding      IP forwarding status
prefix-list     Build a prefix list
rip             Show RIP routes
route           IP routing table
6OS{show-route}show ip rip ?
<cr>
status IP routing protocol process parameters and statistics
6OS{show-route}show ip rip status ?
<cr>
6OS{show-route}show ip rip status
Routing Protocol is "rip"
  Sending updates every 30 seconds with +/-50%, next due in 33 seconds
  Timeout after 180 seconds, garbage collect after 120 seconds
  Outgoing update filter list for all interface is not set
  Incoming update filter list for all interface is not set
  Default redistribution metric is 1
  Redistributing:
  Default version control: send version 2, receive version 2
    Interface      Send  Recv  Key-chain
    eth0_0         2    2
    eth2_0         2    2
  Routing for Networks:
    eth0_0
    eth2_0
  Routing Information Sources:
    Gateway        BadPackets BadRoutes  Distance Last Update
  Distance: (default is 120)

```

Displaying routing protocols status

Displaying RIP status

The `show ip rip status` command displays a summary of RIP protocol status. This is the first thing to do to check whether the RIP session is OK. If RIP protocol is not enabled, nothing will be displayed.

Example:

```

6OS{show-route}show ip rip status
Routing Protocol is "rip"
  Sending updates every 30 seconds with +/-50%, next due in 33 seconds
  Timeout after 180 seconds, garbage collect after 120 seconds
  Outgoing update filter list for all interface is not set
  Incoming update filter list for all interface is not set
  Default redistribution metric is 1
  Redistributing:
  Default version control: send version 2, receive version 2
    Interface      Send  Recv  Key-chain
    eth0_0         2    2
    eth2_0         2    2
  Routing for Networks:
    eth0_0
    eth2_0

```

```
Routing Information Sources:
  Gateway          BadPackets BadRoutes  Distance Last Update
Distance: (default is 120)
```

This status notably lists redistributed protocols, used filters, interfaces on which RIP is activated and networks that should be announced. The most important information in this status display is the list of sending and receiving interfaces: only these interfaces receive and/or send RIP multicast announces. Other interfaces do not run RIP, for any reason (interface is down, interface has no IPv4 address, interface was not detected, interface is configured not to run RIP...)

Displaying RIPng status

The `show ipv6 ripng status` command displays a summary of RIPng protocol status. This is the first thing to do to check whether the RIPng session is OK. If RIPng protocol is not enabled, nothing will be displayed.

Example:

```
6OS{show-route}show ipv6 ripng status
Sending updates every 30 seconds with +/-50%, next due in 12 seconds
Timeout after 180 seconds, garbage collect after 120 seconds
Outgoing update filter list for all interface is not set
Incoming update filter list for all interface is not set
Default redistribution metric is 1
Redistributing:
Default version control: send version 1, receive version 1
  Interface          Send  Recv
  eth0_0              1    1
  eth2_0              1    1
Routing for Networks:
  eth0_0
  eth2_0
Routing Information Sources:
  Gateway          BadPackets BadRoutes  Distance Last Update
  fe80::240:33ff:fe54:b1a0
                                0          0          120      00:00:04
Distance: (default is 120)
```

This status notably lists redistributed protocols, used filters, interfaces on which RIPng is activated and networks that should be announced. The most important information in this status display is the list of sending and receiving interfaces: only these interfaces receive and/or send RIPng multicast announces. Other interfaces do not run RIPng, for any reason (interface is down, interface has no IPv6 link local address, interface was not detected, interface is configured not to run RIPng...)

Displaying OSPF status

The `show ip ospf` command displays a summary of OSPF protocol status. This is the first thing to do to check whether the OSPF session is OK. If OSPF protocol is not enabled, nothing will be displayed.

```
6OS{show-route}show ip ospf
```

Example:

```
6OS{show-route}show ip ospf
OSPF Routing Process, Router ID: 10.23.4.104
Supports only single TOS (TOS0) routes
```

```

This implementation conforms to RFC2328
RFC1583Compatibility flag is disabled
OpaqueCapability flag is disabled
SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
Refresh timer 10 secs
Number of external LSA 0
Number of areas attached to this router: 1

Area ID: 0.0.0.0 (Backbone)
  Number of interfaces in this area: Total: 3, Active: 6
  Number of fully adjacent neighbors in this area: 2
  Area has no authentication
  SPF algorithm executed 3 times
  Number of LSA 4
    
```

Additional information can be obtain with the following commands:

- `show ip ospf border-routers`: list of border routers for this area
- `show ip ospf database`: summary of the database (router and network link states)
- `show ip ospf neighbor`: list of OSPF neighbors

Example:

```
6OS{show-route}show ip ospf database
```

```
OSPF Router with ID (10.23.4.104)
```

```
Router Link States (Area 0.0.0.0)
```

Link ID	ADV Router	Age	Seq#	CkSum	Link count
10.23.4.104	10.23.4.104	1515	0x80000003	0x3868	3
10.24.4.204	10.24.4.204	1518	0x80000003	0x9dc7	3

```
Net Link States (Area 0.0.0.0)
```

Link ID	ADV Router	Age	Seq#	CkSum
10.16.0.197	10.24.4.204	1518	0x80000001	0xae58
10.23.4.204	10.24.4.204	1518	0x80000001	0xe70d

```
6OS{show-route}show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.24.4.204	1	Full/DR	00:00:31	10.23.4.204	eth2_0:10.23.4
.104	0	0	0		
10.24.4.204	1	Full/DR	00:00:31	10.16.0.197	eth1_0:10.16.0
.187	0	0	0		

Displaying OSPFv3 status

As with ospf the most important commands to verify ospf6 operation area

- `show ipv6 ospf6 neighbor`
- `show ipv6 ospfv6 database`

Example:

```
rt1{show-route}show ipv6 ospf6 neighbor
```

RouterID	State/Duration	DR	BDR	I/F[State]
----------	----------------	----	-----	------------

```

10.1.1.2      Full/00:25:36    10.1.1.1    10.1.1.2    eth0_0[DR]
10.1.1.3      Full/00:05:03    10.1.1.1    10.1.1.2    eth0_0[DR]

```

Note that the state must be at **Full**, otherwise it means that the OSPFv3 neighborhood is not correctly formed

Displaying BGP status

To check the TCP connection of BGP for IPv4, type the command `show ip bgp summary` at the show route context.

Example:

```

rt1{show-route}show ip bgp
BGP table version is 0, local router ID is 192.168.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    10.1.1.4          0           0 65520 i
* i                 10.1.1.4          0          100      0 65520 i
* i192.168.0.0     192.168.0.2      0           0 100      0 i
*>                  0.0.0.0          0           0          32768 i

Total number of prefixes 2

```

The various fields, that are displayed, are described in the following list:

- Local router ID --- This id is used to identify the router , it is chosen as the highest ip address configured on an interface. The router id can be manually configured using the command `bgp router-id`
- Status codes :
 - o **s** means the entry is suppressed
 - o **d** means the prefix was dampened (in case dampening is enabled)
 - o ***** means the entry is valid
 - o **>** in case there are multiple entries for this prefix **>** indicates the best one
 - o **i** means that this entry was learnt by interior BGP (in the same AS)
 - o **e** means that this entry was learnt by exterior BGP (different AS)
 - o **?** means that this entry was learnt through redistribution
- Origin codes :
 - o **i** means that the origin of this entry is interior BGP (in the same AS)
 - o **e** means that the origin of this entry is exterior BGP (different AS)
 - o **?** means that the origin of this entry is redistribution
- Network --- This is the prefix entry in the BGP RIB
- Next Hop --- This is the next-hop attribute advertised by BGP in case the next-hop is 0.0.0.0 it means that this prefix is originated locally.
- Metric --- This the Multi-exit Discriminator value (See BGP options)
- LocPrf --- This is the local preference value (See BGP option) by default it is set to 100
- Weight --- This the weight associated to the prefix (See BGP options) by default the received routes is set to 0 while the weight of local routes is 32768

To check the TCP connection of BGP for IPv6, type the command `show ipv6 bgp summary` at the show route context.

Example:

```
rt3{show-route}show ipv6 bgp summary
BGP router identifier 10.1.1.3, local AS number 65530
1 BGP AS-PATH entries
0 BGP community entries

Neighbor      V    AS MsgRcvd MsgSent TblVer  InQ  OutQ  Up/Down
                               State/PfxRcd
2001:500::4  4 65530      42     65     0    0    0 00:30:50    1
3ffe::cc00:2 4 65530      35     45     0    0    0 00:31:09    1

Total number of neighbors 2
```

Displaying the forwarding table and protocols routing tables

Displaying the IPv4 and IPv6 forwarding tables (FIBs)

The `show ip route` command and `show ipv6 route` respectively display the IPv4 FIB (Forwarding Information Table) and IPv6 FIB, that is to say the routes actually used by the IP stack to forward packets.

```
6OS{show-route}show ip route [options]
6OS{show-route}show ipv6 route [options]
```

example1:

```
6OS{show-route}show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, D - DEP, > - selected route, * - FIB route

C>* 10.16.0.0/24 is directly connected, eth1_0
C>* 10.22.4.0/24 is directly connected, eth0_0
C>* 10.23.4.0/24 is directly connected, eth2_0
R>* 10.24.4.0/24 [120/2] via 10.23.4.204, eth2_0, 00:00:27
C>* 127.0.0.0/8 is directly connected, lo0
```

example2:

```
6OS{show-route}show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng, O - OSPFv3,
       B - BGP, D - DEP, N - NAT-PT, > - selected route, * - FIB route.

C>* ::1/128 is directly connected, lo0
C>* 3ffe:304:124:2204::/64 is directly connected, eth0_0
C>* 3ffe:304:124:2304::/64 is directly connected, eth2_0
R>* 3ffe:304:124:2404::/64 [120/2] via fe80::240:33ff:fe54:b1a0, eth2_0, 00:03:11
C * fe80::/64 is directly connected, eth2_0
C * fe80::/64 is directly connected, eth0_0
C>* fe80::/64 is directly connected, lo0
```

Routes in the FIB originate from all activated routing protocols, network interfaces configuration and statically configured routes. Routes displayed by the `show ip route` command are prefixed with codes that identify where the routes come from. The “FIB route” flag means that this route is activated in the forwarding table. The “selected route” flag means that this route may be redistributed to other routing protocols. Routes without one of these 2 flags are not activated in the forwarding table.

Options may be added to the `show ip route` command, to display a subset of the IPv4 FIB:

- `bgp`: routes learnt by BGP
- `connected`: directly connected routes
- `dep`: routes acquired by prefix delegation
- `kernel`: routes added by advanced external modules
- `ospf`: routes acquired by OSPF protocol
- `rip`: routes acquired by RIP
- `static`: static routes
- `A.B.C.D`: a specific route to a host
- `A.B.C.D/M`: a specific route to a network
- `supernets-only`: class-based IPv4 routes

Options may be added to `show ipv6 route` command, to display a subset of the IPv6 FIB:

- `bgp`: routes learnt by BGP
- `connected`: directly connected routes
- `dep`: routes acquired by prefix delegation
- `kernel`: routes added by advanced external modules
- `natpt`: routes added by the NAT-PT function
- `ospf6`: routes acquired by OSPFv3 protocol
- `ripng`: routes acquired by RIPng
- `static`: static routes
- `X:X::X:X`: a specific route to a host
- `X:X::X:X/M`: a specific route to a network

Displaying the RIP routing table (RIP RIB)

The RIP RIB (Routing Information Base) holds IPv4 routes learnt by the RIP protocol. These routes are reported to the IPv4 FIB, which will build a forwarding table by choosing among routes gathered from all routing protocols.

The RIP RIB can be dumped by using the `show ip rip` command.

Example:

```
6OS{show-route}show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
       D - DEP
Sub-codes:
          (n) - normal, (s) - static, (d) - default, (r) - redistribute,
          (i) - interface

          Network          Next Hop          Metric From          Tag Time
C(i) 10.22.4.0/24         0.0.0.0           1 self              0
C(i) 10.23.4.0/24         0.0.0.0           1 self              0
R(n) 10.24.4.0/24         10.23.4.204       2 10.23.4.204       0 02:34
```

Displaying the RIPng routing table (RIPng RIB)

The RIPng RIB (Routing Information Base) holds IPv6 routes learnt by the RIPng protocol. These routes are reported to the IPv6 FIB, which will build a forwarding table by choosing among routes gathered from all routing protocols.

The RIPng RIB can be dumped by using the `show ipv6 ripng` command.

Example:

```
6OS{show-route}show ipv6 ripng
Codes: R - RIPng, C - connected, S - Static, O - OSPF, B - BGP
       D - DEP, N - NAT-PT
Sub-codes:
        (n) - normal, (s) - static, (d) - default, (r) - redistribute,
        (i) - interface, (a/S) - aggregated/Suppressed

      Network      Next Hop                Via      Metric Tag Time
C(i) 3ffe:304:124:2204::/64
      :
      self          1      0
C(i) 3ffe:304:124:2304::/64
      :
      self          1      0
R(n) 3ffe:304:124:2404::/64
      fe80::240:33ff:fe54:b1a0 eth2_0      2      0 02:34
```

Displaying the OSPF routing table (OSPF RIB)

The OSPF RIB (Routing Information Base) holds IPv4 routes learnt by the OSPF protocol. These routes are reported to the IPv4 FIB, which will build a forwarding table by choosing among routes gathered from all routing protocols.

The OSPF RIB can be dumped by using the `show ip ospf` command.

Example:

```
6OS{show-route}show ip ospf route
===== OSPF network routing table =====
N   10.16.0.0/24      [10] area: 0.0.0.0
      directly attached to eth1_0
N   10.22.4.0/24     [10] area: 0.0.0.0
      directly attached to eth0_0
N   10.23.4.0/24     [10] area: 0.0.0.0
      directly attached to eth2_0
N   10.24.4.0/24     [20] area: 0.0.0.0
      via 10.23.4.204, eth2_0
      via 10.16.0.197, eth1_0

===== OSPF router routing table =====

===== OSPF external routing table =====
```

Displaying the OSPFv3 routing table (OSPFv3 RIB)

The OSPFv3 RIB (Routing Information Base) holds IPv6 routes learnt by the OSPFv3 protocol. These routes are reported to the IPv6 FIB, which will build a forwarding table by choosing among routes gathered from all routing protocols.

The OSPFv3 RIB can be dumped by using the `show ipv6 ospf6` command.

Example:

```
rt1{show-route}show ipv6 ospf6 route
```

```

      Destination          Gateway                I/F
*N Ia 2001:500::/64      ::                    eth0_0    00:27:54
*N Ia 3ffe:2::/64       fe80::209:c0ff:fe30:4046 eth0_0    00:21:41
*N Ia 3ffe:3::/64       fe80::209:c0ff:fe30:4358 eth0_0    00:07:21
=====
Route: 3 Path: 3 Redundant: 0

```

Displaying the BGP IPv4 and IPv6 routing tables (BGP RIBs)

The BGP RIBs (Routing Information Bases) hold the IPv4 and IPv6 routes learnt by the BGP protocol. These routes are reported to the IPv4 and IPv6 FIB, which will build a forwarding table by choosing among routes gathered from all routing protocols.

The IPv4 BGP RIB can be dumped by using the `show ip bgp` command.

The IPv6 BGP RIB can be dumped by using the `show ipv6 bgp` command.

Example:

```

6OS{show-route}show ip bgp
BGP table version is 0, local router ID is 192.168.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    10.1.1.4          0             0 65520 i
* i                 10.1.1.4          0      100        0 65520 i
* i192.168.0.0     192.168.0.2      0      100        0 i
*>                  0.0.0.0          0             32768 i

Total number of prefixes 2

```

The various fields, that are displayed, are described in the following list:

- Local router ID --- This id is used to identify the router, it is chosen as the highest ip address configured on an interface. The router id can be manually configured using the command `bgp router-id`
- Status codes :
 - o **s** means the entry is suppressed
 - o **d** means the prefix was dampened (in case dampening is enabled)
 - o ***** means the entry is valid
 - o **>** in case there are multiple entries for this prefix **>** indicates the best one
 - o **i** means that this entry was learned by interior BGP (in the same AS)
 - o **e** means that this entry was learned by exterior BGP (different AS)
 - o **?** means that this entry was learned through redistribution
- Origin codes :
 - o **i** means that the origin of this entry is interior BGP (in the same AS)
 - o **e** means that the origin of this entry is exterior BGP (different AS)
 - o **?** means that the origin of this entry is redistribution
- Network --- This is the prefix entry in the BGP RIB
- Next Hop --- This is the next-hop attribute advertised by BGP in case the next-hop is 0.0.0.0 it means that this prefix is originated locally.
- Metric --- This the Multi-exit Discriminator value (See BGP options)
- LocPrf --- This is the local preference value (See BGP option) by default it is set to 100
- Weight --- This the weight associated to the prefix (See BGP options) by default the received routes is set to 0 while the weight of local routes is 32768

- **Weight** --- This the weight associated to the prefix (See BGP options) by default the received routes is set to 0 while the weight of local routes is 32768

```
6OS{show-route}show ipv6 bgp
BGP table version is 0, local router ID is 10.1.1.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
* i2001:500::/64	2001:500::3	0	100	0	i
* i	3ffe::cc00:1	0	100	0	i
*>	::	0		32768	i

Total number of prefixes 1

PPP status display

Once a PPP negotiation with a remote PPP equipment has succeeded, a logical network interface is created. All physical and logical network interfaces can be displayed by invoking the `show interface` command in the root context.

Example:

```
6OS{}show interface
...
ppp1: flags=88d1<UP, POINTOPOINT, RUNNING, NOARP, SIMPLEX, MULTICAST> mtu 1492
      user: foobar@6wind.myisp.net
      inet 10.132.0.54 --> 10.132.4.104 netmask 0xffffffff
ppp2: flags=88d1<UP, POINTOPOINT, RUNNING, NOARP, SIMPLEX, MULTICAST> mtu 1492
      user: foobar@6wind.myisp.net
      inet 10.132.0.54 --> 10.132.4.103 netmask 0xffffffff
```

This logical interface disappears when the PPP connection is closed.

IKE status display

The IKE status may be checked by displaying ISAKMP and IPsec SAs negotiations. The `show ike` command is provided for this purpose.

```
6OS{}show ike {isakmp|ipsec} [{active|inactive}]
```

Where:

- **isakmp** requests that ISAKMP (IKE phase 1) negotiations should be displayed
- **ipsec** requests that IPsec (IKE phase 2) negotiations should be displayed
- **active** use this optional parameter to only display active SA negotiations
- **inactive** use this optional parameter to only display inactive SA negotiations

Active SA negotiations are ISAKMP or IPsec SA negotiations in the **established** state, i.e. that have successfully generated SAs. On the contrary, inactive SA negotiations are ISAKMP or IPsec SA negotiations that are still in progress or whose SAs have expired.

Example:

```
6OS{}show ike isakmp
=== ISAKMP negotiations: =====
```

```
VPN 10.23.4.104[500] 10.23.4.205[500]
  exchange type      main mode
  side               initiator
  state              1st msg sent
  initiator cookie   2c6991b3fb7f5ef7
  responder cookie   0000000000000000
  achieved phase2    0
VPN 10.23.4.104[500] 10.23.4.204[500]
  exchange type      main mode
  side               responder
  state              SA established
  initiator cookie   f0f9f99d1a9eecbc
  responder cookie   068b5070dc35f50c
  creation date      2004-08-25 18:36:53
  achieved phase2    2
```

In the above example, an ISAKMP negotiation is in progress between the 6WINDGate (10.23.4.104 port 500) and a remote IPsec device (10.23.4.205 port 500). The 6WINDGate is the initiator of the negotiation and has sent the first message of a main mode exchange.

A second ISAKMP session was established at 18h36 with a remote IPsec device (10.23.4.204 port 500). The 6WINDGate was the responder of the negotiation, and used main mode. Two successful quick mode (phase 2) negotiations were then established with the remote IPsec device.

The `show ike isakmp active` command would only display the second ISAKMP negotiation. The `show ike isakmp inactive` command would only display the first ISAKMP negotiation.

```
6OS{ }show ike ipsec
=== IPsec SAs negotiations: =====
Tunnel 10.22.4.0/24[0] 10.25.4.0/24[0] 0
  message ID          45391574
  side                initiator
  state               1st msg sent
  ph1 local peer      10.23.4.104[500]
  ph1 remote peer     10.23.4.204[500]
  ph1 initiator cookie f0f9f99d1a9eecbc
  ph1 responder cookie 068b5070dc35f50c
Tunnel 10.23.4.104[0] 10.23.4.204[0] 0
  message ID          77bc9cd3
  side                responder
  state               SAs established
Tunnel 10.22.4.0/24[0] 10.24.4.0/24[0] 0
  message ID          abfb77fa
  side                initiator
  state               SAs established
  ph1 local peer      10.23.4.104[500]
  ph1 remote peer     10.23.4.204[500]
  ph1 initiator cookie f0f9f99d1a9eecbc
  ph1 responder cookie 068b5070dc35f50c
```

In the above example, an IPsec negotiation is in progress between the 6WINDGate (10.23.4.104 port 500) and a remote IPsec device (10.23.4.204 port 500), to establish IPsec SAs for tunnel 10.22.4.0/24[0] 10.24.4.0/24[0] 0 (any protocol from network 1.22.4.0/24 to network 10.24.4.0/24). The 6WINDGate is the initiator of the negotiation and has sent the first message of a main mode exchange.

Two IPsec SA negotiations have been established for traffic 10.23.4.104[0] 10.23.4.204[0] 0 (the quick mode exchange was initiated by the remote IPsec device), and for traffic 10.22.4.0/24[0] 10.24.4.0/24[0] 0 (the quick mode exchange was initiated by the 6WINDGate). The parent ISAKMP negotiation for the later is identified by the IPsec peer addresses (10.23.4.104 and 10.23.4.204) and the initiator and responder cookies (f0f9f99d1a9eeccb and 068b5070dc35f50c).

The `show ike ipsec active` command would only display the two last negotiations.

The `show ike ipsec inactive` command would only display the first negotiation.

IPsec status display

The IPsec status may be checked by displaying IPsec SAs. The `show ipsec sa` command is provided for this purpose.

```
6OS{ }show ipsec sa [{ike|static}]
```

Where:

- `ike` is an optional parameter to only display IPsec SA generated by the IKE protocol
- `static` is an optional parameter to only display manually configured IPsec SAs.

Example:

```
6OS{ }show ipsec sa ike
source=10.23.4.104 destination=10.23.4.204
  protocol=esp mode=any spi=164079602(0x09c7a7f2) reqid=16455(0x00004047)
  encr-algo=3des-cbc
  encr-key=8b685e84f1809f4a37f4ced957b90b66ffa7d5e87aef6e9d
  auth-algo=hmac-sha1
  auth-key=98b299c797b85d2e8cd5d46c7a680ebd183b3aa1
  replay-window=4 flags=0x00001000 state=mature seq=4 pid=5402
  created=2004-08-25/18:36:53 current=2004-08-25/18:52:58 elapsed=965(s)
  hard-lifetime=3600(s) expiration=2004-08-25/19:36:53
  soft-lifetime=2880(s) renewal=2004-08-25/19:24:53
  last-use=2004-08-25/18:36:56
  bytes-processed=408 hard-lifebyte=1048576000 soft-lifebyte=838860800
source=10.23.4.204 destination=10.23.4.104
  protocol=esp mode=any spi=56401068(0x035c9cac) reqid=16456(0x00004048)
  encr-algo=3des-cbc
  encr-key=911662222700cc0d6086d83da7e43481edae4a56c5daefb4
  auth-algo=hmac-sha1
  auth-key=e80bcc4c29fef3a69de7102db4a6c24f9a1ac3c6
  replay-window=4 flags=0x00001000 state=mature seq=3 pid=5402
  created=2004-08-25/18:36:53 current=2004-08-25/18:52:58 elapsed=965(s)
  hard-lifetime=3600(s) expiration=2004-08-25/19:36:53
  soft-lifetime=2880(s) renewal=2004-08-25/19:24:53
  last-use=2004-08-25/18:36:56
  bytes-processed=252 hard-lifebyte=1048576000 soft-lifebyte=838860800
source=10.23.4.104 destination=10.23.4.204
  protocol=esp mode=any spi=40227391(0x0265d23f) reqid=16457(0x00004049)
  encr-algo=3des-cbc
  encr-key=d4904763f3bac8e7f1d29f97ee7e560789e7076e067c8842
  auth-algo=hmac-sha1
  auth-key=1d99762d2026b2aeeb15a727844aaedc0b5f8d1a
  replay-window=4 flags=0x00001000 state=mature seq=2 pid=5402
  created=2004-08-25/18:36:53 current=2004-08-25/18:52:58 elapsed=965(s)
```

```
hard-lifetime=3600(s) expiration=2004-08-25/19:36:53
soft-lifetime=2880(s) renewal=2004-08-25/19:24:53
last-use=2004-08-25/18:36:56
bytes-processed=408 hard-lifebyte=1048576000 soft-lifebyte=838860800
source=10.23.4.204 destination=10.23.4.104
protocol=esp mode=any spi=93517238(0x0592f5b6) reqid=16458(0x0000404a)
encr-algo=3des-cbc
encr-key=bb5a9950af3f639c4960bf674cd1bbe0cdd4d07b4cbf34fc
auth-algo=hmac-sha1
auth-key=c07ca36be44b925645580cc0ff6ea7af2299c00d
replay-window=4 flags=0x00001000 state=mature seq=1 pid=5402
created=2004-08-25/18:36:53 current=2004-08-25/18:52:58 elapsed=965(s)
hard-lifetime=3600(s) expiration=2004-08-25/19:36:53
soft-lifetime=2880(s) renewal=2004-08-25/19:24:53
last-use=2004-08-25/18:36:56
bytes-processed=252 hard-lifebyte=1048576000 soft-lifebyte=838860800
```

Here is a description of the displayed information:

- **source** source address of the SA
- **destination** destination address of the SA
- **protocol** IPsec protocol (ESP or AH)
- **mode** IPsec mode (transport or tunnel)
- **spi** security parameters index
- **reqid** internal unique identifier of the security policy that triggered SA creation
- **encr-algo** encryption algorithm
- **encr-key** encryption key, in hexadecimal format
- **auth-algo** authentication algorithm
- **auth-key** authentication key, in hexadecimal format
- **replay-window**
size of the replay window. 0 if replay protection is disabled
- **flags, seq, state, seq, pid**
internal values. If the state is different from mature, then the SA cannot be used yet
- **created** date of creation
- **current** current time
- **elapsed** time elapsed since the SA was created
- **hard-lifetime**
when this time has elapsed since SA creation, the SA will expire. If set to 0, the SA will not expire based on elapsed time.
- **soft-lifetime**
when this time has elapsed since SA creation, a new SA will be negotiated to replace it, before it reaches its hard lifetime. If set to 0, the SA will not be renegotiated based on elapsed time
- **expiration** date when the SA will reach its hard lifetime and expire
- **renewal** date when the SA will reach its soft lifetime and a new negotiation will be requested
- **last-use** date when the SA was last used by the IPsec module
- **bytes-processed**
volume of data that was processed by the IPsec module with this SA

ESP output histogram:

3des-cbc: 15

IPv6

```
0 inbound packets processed successfully
0 inbound packets violated process security policy
0 inbound packets with no SA available
0 invalid inbound packets
0 inbound packets failed due to insufficient memory
0 inbound packets failed getting SPI
0 inbound packets failed on AH replay check
0 inbound packets failed on ESP replay check
0 inbound packets considered authentic
0 inbound packets failed on authentication
0 outbound packets processed successfully
0 outbound packets violated process security policy
0 outbound packets with no SA available
0 invalid outbound packets
0 outbound packets failed due to insufficient memory
```

Chapter 30 Troubleshooting the 6WINDGate – Using logging facilities

Configuring the logging service

Introduction

The logging service is provided in order to troubleshoot or monitor events occurring on the 6WINDGate. Logging is provided via the standard syslog service, which enables to store and display log messages locally or on a remote syslog server.

Log messages may be generated for various reasons. They are all tagged with a severity level, so that an administrator may filter messages based on their criticality.

These log messages can either be stored in a local session or sent to a remote server via syslog. Messages stored in a local session can also be later exported as a text file using an `export` command.

To store messages remotely, the 6WINDGate implements a syslog client. The transfer between the 6WIND equipment and the remote syslog server can use IPv4 or IPv6 protocols.

Configuration overview

Logs configuration is divided in two levels:

- “log sessions” define where log messages should be stored and optionally set storage size limits. They are configured in the root context,
- then, logging is separately enabled or disabled in each function context. If logging is enabled, messages may be filtered based on their severity level. Messages below a configured severity are discarded.

Configuring logging

Configuring log sessions

A log session defines where log messages should be stored when they are sent to this session. Log messages can be stored in a local session or sent to a remote syslog server via the `logsession` command. Up to 9 log sessions may be defined, named `session1`, `session2`, ..., `session9`. Log sessions in configured in the root context, they are consequently common to all configurations.

To configure log sessions, use the following commands.

```
6OS{}logsession sessionX local size unit
```

or


```
6OS{ } logsession sessionX remote address
```

where:

- *sessionX* is the session name, *x* ranging from 1 to 9.
- **local** indicates that messages are redirected to a local log session.
- **remote** indicates that messages are redirected to a remote syslog server.
- *size* and *unit* (**B**, **KB** or **MB**) set the maximum size of the local log session. When the log session reaches this limit, its content is deleted. The minimum size is 2 KB; the maximum total size is platform specific.
- *address* is the IPv4 or IPv6 address of the syslog server.

Example:

```
6OS{myconfig} logsession session1 local 20 KB
6OS{myconfig} logsession session2 remote 10.0.0.193
6OS{myconfig} logsession session3 remote 3ffe:304:124::1
```

 **Note** When configuring a remote log session, if IP filtering is enabled, make sure that syslog traffic to the server is allowed by filtering rules. Syslog servers listen on UDP port 514.


Deleting a Log Session

To delete a log session, use the following command:

```
6OS{ } delete logsession sessionX
```

where:

- *sessionX* is the session name, *x* ranging from 1 to 9.

 **Note** Log session modifications are taken into account during the next **apply** or **addrunning**.

Configuring log messages to record

Most of the 6WINDGate functions provide a `log` command, to configure which events should be reported and where they should be sent. Logs settings are stored in configurations and can be edited in each function context.

The prototype for this command is:

```
log function sessionX [severity]
```

where:

- *function* is the function whose logs are configured,
- *sessionX* is the session to which logs will be sent,
- *severity* is an optional severity level below which log messages will be discarded.

Example (routing context):

```
6OS{myconfig-rtg} log rip session1 error
```

This command will log all RIP log messages of severity level greater or equal to LOG_ERR to *session1*, if *session1* is defined.

If the optional *level* parameter is not provided, **notice** is assumed, which stands for syslog LOG_NOTICE severity level.

Disabling log messages

Logging for a function can be disabled by using the delete log function. Since the log command can be used several times to send logs to several sessions, the session name must be provided in a delete log command.

```
delete log function sessionX
```

Example (routing context):

```
6OS{myconfig-rtg}delete log rip session1
```

System Logging Severity Levels

The event logging system provides eight severity levels. The highest level of message is level 0, the lowest level is level 7.

Level	Keyword	Description	Syslog severity
0	emergency	Panic or other conditions that cause the system to become unusable.	LOG_EMERG
1	alert	Immediate action is needed. (correct a corrupted database).	LOG_ALERT
2	critical	Critical conditions exist.	LOG_CRIT
3	error	Error conditions exist.	LOG_ERR
4	warning	Warning conditions exist.	LOG_WARNING
5	notice	Conditions which are not errors but confirm manipulations.	LOG_NOTICE
6	info	Informational messages.	LOG_INFO
7	debug	Debugging messages.	LOG_DEBUG

Displaying logging information

Displaying logging configuration

To display information about logging configuration, use the `display` command in the relevant context.

Example:

To display log sessions:

```
6OS{}display logsession
<name> [local <size> <size unit>] [remote <address>]

# LOGSESSIONS
session1 local 20 KB
session2 remote 10.0.0.193
```

To display logging configuration for `gen` context:

```
6OS{myconfig-gen}display
...

# LOG
log system session1
log system session2 error
```

According to the former definition of `session1` and `session2`, all system log messages will be stored in a local session, and log messages whose severity is greater or equal to `LOG_ERR` will be sent to a remote syslog server located at 10.0.0.193 IPv4 address.

To display logging configuration for `rtg` context:

```
6OS{myconfig-rtg}display
...

# LOG
log rip session1 notice
```

Watching locally stored log messages

To display log messages sent by the 6WINDGate to a local log session, use the `show log` command, in the root context.

Example:

```
6OS{ }show log session1
2004/05/15 14:25:38 SYSTEM-CRIT: ipfw: 59999 Deny UDP 10.16.0.4:1025
10.16.0.113:53 out via fxp1
2004/05/15 14:26:35 SYSTEM-CRIT: ipfw: 59999 Deny UDP 10.16.0.119:137
10.16.0.255:137 in via fxp1
2004/05/15 14:26:40 SYSTEM-CRIT: ipfw: 59999 Deny ICMP:8.0 10.16.0.4 10.16.0.1
out via fxp1
```

Exporting a log session

To export a text file containing log messages of a local log session, use the `export log` command, in the root context:

```
6OS{ }export log sessionX URL
```

`tftp`, `ftp` or `scp` protocols can be used to export log sessions.

Example:

```
6OS{ }export log session3 ftp://admin:pswd6@10.0.0.1//log_arch/session3
6OS{ }export log session3 ftp://admin:pswd6@[3ffe:304:124::2]//log_arch/session3
6OS{ }export log session2 tftp://10.0.0.1/rem_session2
6OS{ }export log session1 scp://admin@logserver.domain.com/export_dir/rem_session1
```

Note 1 The remote file name must be provided in the URL when using `ftp` or `tftp`. The remote file name is optional with `scp`.

Note 2 Most TFTP server implementations require that an empty file be created on the remote server (with reading and writing privileges) before the log session can be exported by TFTP.

Routing log messages

Routing functions are likely to generate a great amount of log messages. Therefore, routing log messages may be configured very precisely.

Configuration overview

Routing log messages are configured at 2 levels. First the administrator can choose which protocol should send logs and filter them by severity (for instance RIP can send logs with severity \geq LOG_ERR). Then, for each protocol, the administrator can choose extra information that should be reported, in addition to general routing log messages (for instance RIP should report all protocol events and sent or received RIP packets).

Routing log messages configuration

Configuring protocols that should send logs

The administrator can activate logs for the FIB and up to 5 routing protocols:

- `log fib`: message logs related to the FIB (Forwarding Information Table)
- `log rip`: message logs related to the RIP protocol (Routing Internet Protocol)
- `log ripng`: message logs related to the RIPng protocol (RIP for IPv6)
- `log bgp`: message logs related to the BGP protocol (Border Gateway Protocol)
- `log ospf`: message logs related to the OSPF protocol (Open Shortest Path First)
- `log ospf6`: message logs related to the OSPFv3 protocol (OSPF for IPv6)

These logs are respectively activated by the following commands, in the `rtg` context of a configuration.

```
6OS{myconfig-rtg}log fib sessionX [severity]
6OS{myconfig-rtg}log rip sessionX [severity]
6OS{myconfig-rtg}log ripng sessionX [severity]
6OS{myconfig-rtg}log bgp sessionX [severity]
6OS{myconfig-rtg}log ospf sessionX [severity]
6OS{myconfig-rtg}log ospf6 sessionX [severity]
```

The optional parameter `severity` enables to specify a minimal severity level, under which log messages will be discarded. By default, it is set to `notice`.

By default, only important information is reported, which has little influence on the 6WINDGate performance. Most log message severities are greater or equal to LOG_WARNING.

Requesting protocols to report extra information

For each protocol, additional logs can be requested. These logs can however alter the 6WINDGate performance if too many messages are generated. These messages are configured in the `dynamic` subcontext.

- `debug rip events`: report all RIP events
- `debug rip memory`: report RIP memory usage
- `debug rip packet`: report received or sent RIP packets
- `debug ripng events`: report all RIPng events
- `debug ripng memory`: report RIPng memory usage
- `debug ripng packet`: report received or sent RIPng packets

- `debug bgp events`: report all BGP events
- `debug bgp filters`: report filtering decisions (loop avoidance, filtering policies...)
- `debug bgp fsm`: report states of the BPG Finite State Machine
- `debug bgp keepalives`: report received or sent BGP keepalive packets
- `debug bgp updates`: report received or sent BGP update packets

Example:

```
6OS{myconfig-rtg-dynamic}debug rip events
6OS{myconfig-rtg-dynamic}debug bgp fsm
6OS{myconfig-rtg-dynamic}debug bgp updates
```

PPP log messages

PPP functions are likely to generate a great amount of log messages. Therefore, PPP log messages may be configured very precisely.

Configuration overview

PPP log messages can be filtered according to the network layer they refer to, instead of their severity. PPP network layers that should report log messages are listed by the administrator. The administrator may then enable or disable logs and send them to one or more log sessions.

PPP log messages configuration

Activating PPP logs

The administrator can globally activate logs for a PPP interface. These logs are activated by the following command, in the `ppp_serverX` context of a configuration. Several `log ppp` commands may be supplied:

```
6OS{myconfig-ppp_serverX}log ppp sessionY
```

Where

- `sessionY` is the log session to which these logs should be sent.

Example:

```
6OS{myconfig-ppp_serverX}log ppp session1
6OS{myconfig-ppp_serverX}log ppp session2
```

Configuring PPP network layers that should send logs

The administrator can filter PPP network layers that should send logs, by using the `debug` statement in the `pppX` context of a configuration.

- Network layers can be configured in a single command line, as a quoted list:

```
6OS{myconfig-ppp_serverX}debug ppp "layerY layerZ ..."
```

- Or iteratively, by adding or removing a layer:

```
6OS{myconfig-ppp_serverX}debug ppp layerX
```

```
6OS{myconfig-ppp_serverX}delete debug ppp layerX
```

- All network layers can be removed in a single command line:


```
6OS{myconfig-ppp_serverX}debug ppp none
```


Example:

```
6OS{myconfig-ppp_server0}debug ppp "LCP AUTH ECHO FSM"
6OS{myconfig-ppp_server0}debug ppp IPCP
6OS{myconfig-ppp_server0}delete debug ppp FSM
```

Here follows the list of supported network layer keywords:

- **BUND**: bundle events
- **LINK**: link events
- **LCP**: LCP (Link Control Protocol) events and negotiation
- **AUTH**: link authentication events
- **IPCP**: IPCP (IP Control Protocol) events and negotiation
- **IPV6CP**: ipv6cp events and negotiation
- **CCP**: CCP (Compression Control Protocol) events and negotiation
- **CCP2**: CCP additional debugging output
- **CCP3**: CCP complete packet dumps
- **ECP**: ECP events and negotiation
- **ECP2**: ECPextra debugging output
- **FSM**: all FSM (Finite State Machine) events (except echo & reset)
- **ECHO**: echo/reply events for all automata
- **PHYS**: physical layer events
- **CHAT**: modem chat script
- **CHAT2**: chat script extra debugging output
- **IFACE**: IP interface and route management
- **FRAME**: dump all incoming & outgoing frames
- **PPTP**: PPTP high level events
- **PPTP2**: PPTP more detailed events
- **PPTP3**: PPTP packet dumps
- **RADIUS**: RADIUS authentication events
- **L2TP**: L2TP (Layer2 Tunneling Protocol) high level events
- **L2TP2**: L2TP more detailed events
- **L2TP3**: L2TP packet dumps

 **Note 1** Due to the relative verbosity of PPP logs, it is recommended to limit network layers that should send logs, and to reserve PPP logging to debugging needs.

 **Note 2** All network layers are not available on all SixOS versions.

IPsec log messages

The IPsec dynamic key exchange protocol, IKE, is sometimes difficult to configure. Log messages are a very useful tool to understand why an IKE connection cannot be established, and to make out how to tune the configuration. Log messages also enable to trace secure connections establishment.

Configuration overview

All IKE log messages are sent at the debug severity level.

IKE log messages configuration

Activating IKE logs

The administrator can globally activate IKE logs. These logs are activated by the following command, in the `sec` context of a configuration. Several `log ike` commands may be supplied:

```
6OS{myconfig-ppp_serverX}log ike sessionY [severity]
```

Where

- `sessionY` is the log session to which these logs should be sent.
- `severity` is an optional severity level below which log messages will be discarded.

Example:

```
6OS{myconfig-ppp_serverX}log ppp session1 debug  
6OS{myconfig-ppp_serverX}log ppp session2 debug
```

Chapter 31 Troubleshooting the 6WINDGate – Using traffic monitoring

Activating the traffic monitoring

Introduction

The traffic monitoring is provided in order to display and record the traffic on one interface of the 6WINDGate.

This function enables to display or record the traffic, choose the verbosity level and apply a filter.

Displaying the traffic

The syntax of the `show traffic` command for displaying is the following :

```
6OS{myconfig-rtg}show traffic INTERFACE [count N] [verbose V] [proto (all|PROTO)]
[without PROTO] [pcap "FILTER"] [pager]
```

Where

- `count N` specifies the number of packets to dump. The capture stops when the amount is reached.
- `verbose V` specifies the verbosity level. When the level is higher, the description of the packet is more detailed.
- `proto PROTO` specifies the prototype of the packets that are displayed (icmp, tcp, ip6, ...). By default all prototypes are displayed (proto all).
- `without PROTO` specifies the prototype of packets that should be excluded.
- `pcap "FILTER"` specifies a filter in pcap syntax.
- `pager` the display is sent to a pager

Example:

```
6OS{ }show traffic eth2_0 count 10
tcpdump6: listening on ed0
15:27:49.612108 arp who-has 10.23.3.103 tell 10.23.3.203
15:27:49.612218 arp reply 10.23.3.103 is-at 52:54:40:20:e:53
15:27:49.612619 10.23.3.203 > 10.23.3.103: icmp: echo request
15:27:49.612814 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:27:50.613186 10.23.3.203 > 10.23.3.103: icmp: echo request
15:27:50.613377 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:27:51.613971 10.23.3.203 > 10.23.3.103: icmp: echo request
15:27:51.614167 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:27:52.615750 10.23.3.203 > 10.23.3.103: icmp: echo request
```

```
15:27:52.615935 10.23.3.103 > 10.23.3.203: icmp: echo reply
10 packets received by filter
0 packets dropped by kernel
6OS{}
6OS{}show traffic eth2_0 count 10 verbose 2
tcpdump6: listening on ed0
15:30:05.141649 10.23.3.203 > 10.23.3.103: icmp: echo request (ttl 64, id 1786,
len 84)
15:30:05.141897 10.23.3.103 > 10.23.3.203: icmp: echo reply (ttl 255, id 5, len
84)
15:30:06.142571 10.23.3.203 > 10.23.3.103: icmp: echo request (ttl 64, id 1787,
len 84)
15:30:06.142765 10.23.3.103 > 10.23.3.203: icmp: echo reply (ttl 255, id 6, len
84)
15:30:07.143337 10.23.3.203 > 10.23.3.103: icmp: echo request (ttl 64, id 1788,
len 84)
15:30:07.143522 10.23.3.103 > 10.23.3.203: icmp: echo reply (ttl 255, id 7, len
84)
15:30:08.145129 10.23.3.203 > 10.23.3.103: icmp: echo request (ttl 64, id 1789,
len 84)
15:30:08.145314 10.23.3.103 > 10.23.3.203: icmp: echo reply (ttl 255, id 8, len
84)
15:30:14.753615 10.23.3.203 > 10.23.3.103: icmp: echo request (ttl 64, id 1790,
len 84)
15:30:14.753804 10.23.3.103 > 10.23.3.203: icmp: echo reply (ttl 255, id 9, len
84)
10 packets received by filter
0 packets dropped by kernel
6OS{}
6OS{}show traffic eth2_0 without icmp
tcpdump6: listening on ed0
15:34:51.233760 arp who-has 10.23.3.104 (0:0:0:0:1) tell 10.23.3.203
15:34:52.234090 arp who-has 10.23.3.104 (2e:2f:30:31:32:33) tell 10.23.3.203
15:34:53.234861 arp who-has 10.23.3.104 (2e:2f:30:31:32:33) tell 10.23.3.203
15:34:54.235662 arp who-has 10.23.3.104 (2e:2f:30:31:32:33) tell 10.23.3.203
15:34:55.236443 arp who-has 10.23.3.104 (2e:2f:30:31:32:33) tell 10.23.3.203
^C
19 packets received by filter
0 packets dropped by kernel
6OS{}
6OS{}show traffic eth2_0 pcap "src 10.23.3.203"
tcpdump6: listening on ed0
15:39:46.211741 10.23.3.203 > 10.23.3.103: icmp: echo request
15:39:47.212901 10.23.3.203 > 10.23.3.103: icmp: echo request
15:39:48.213685 10.23.3.203 > 10.23.3.103: icmp: echo request
15:39:49.215467 10.23.3.203 > 10.23.3.103: icmp: echo request
15:39:50.217266 10.23.3.203 > 10.23.3.103: icmp: echo request
15:39:51.219043 10.23.3.203 > 10.23.3.103: icmp: echo request
15:39:52.219835 10.23.3.203 > 10.23.3.103: icmp: echo request
15:39:53.221624 10.23.3.203 > 10.23.3.103: icmp: echo request
^C
16 packets received by filter
0 packets dropped by kernel
6OS{}
```

The capture stops when the specified amount of packets is reached or when a ctrl-C is sent.

Recording the traffic

The syntax of the `show traffic` command for recording is the following :

```
6OS{myconfig-rtg}show traffic INTERFACE record FILE [count N] [verbose V] [proto
(all|PROTO)] [without PROTO] [pcap "FILTER"] [pager] [background]
```

Where

- `record FILE` specifies the file name for the capture.
- `background` indicates that the capture is done in background. Any command of the CLI can be entered once the capture is launched.

Stopping a capture

A capture stops when the specified amount of packets is reached.

When the capture is running in foreground a ctrl-C stops the capture

When the capture is running in background, use the `show traffic stop` command to stop the capture

Example:

```
6OS{}show traffic eth2_0 record myfile.cap background
6OS{}tcpdump6: listening on ed0

6OS{}show traffic stop

22 packets received by filter
0 packets dropped by kernel
6OS{}
6OS{}show traffic eth2_0 count 10
tcpdump6: listening on ed0
15:27:49.612108 arp who-has 10.23.3.103 tell 10.23.3.203
15:27:49.612218 arp reply 10.23.3.103 is-at 52:54:40:20:e:53
15:27:49.612619 10.23.3.203 > 10.23.3.103: icmp: echo request
15:27:49.612814 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:27:50.613186 10.23.3.203 > 10.23.3.103: icmp: echo request
15:27:50.613377 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:27:51.613971 10.23.3.203 > 10.23.3.103: icmp: echo request
15:27:51.614167 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:27:52.615750 10.23.3.203 > 10.23.3.103: icmp: echo request
15:27:52.615935 10.23.3.103 > 10.23.3.203: icmp: echo reply
10 packets received by filter
0 packets dropped by kernel
6OS{}
```

Reading a capture file

The syntax of the `show traffic` command for reading a capture file is the following :

```
6OS{myconfig-rtg}show traffic file FILE [count N] [verbose V] [proto (all|PROTO)]
[without PROTO] [tcpdump "FILTER"]
```

Where

- `file FILE` specifies the file name for the capture.
- `tcpdump "FILTER"` specifies a filter with tcpdump syntax.

Example:

```
6OS{}show traffic file myfile.cap tcpdump "src 10.23.3.103"
```

```
15:45:07.174227 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:08.175339 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:09.176131 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:10.176911 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:11.178697 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:12.180515 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:13.181270 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:14.183067 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:15.184846 10.23.3.103 > 10.23.3.203: icmp: echo reply
15:45:16.186643 10.23.3.103 > 10.23.3.203: icmp: echo reply
6OS{}
```

Managing the capture files

The 6WINDGate provides some managing functions for the capture files, i.e. displaying, deleting, exporting and importing

Example:

```
6OS{display traffic-file
myfile.cap
6OS{
6OS{export traffic-file myfile.cap ftp://user:passwd@host/path/destfile.cap
exporting to file ftp://user:passwd@host/path/destfile.cap
100% |*****| 2264 00:00 ETA
6OS{delete traffic-file myfile.cap
6OS{display traffic-file
6OS{
6OS{import traffic-file ftp://user:passwd@host/path/srcfile.cap myfile.cap
importing file ftp://user:passwd@host/path/srcfile.cap
100% |*****| 2264 00:00 ETA
6OS{display traffic-file
myfile.cap
6OS{}
```

Appendix A List of Acronyms

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
AF	Assured Forwarding
AH	Authentication Header
ARP	Address Resolution Protocol
ASN1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
BA	Behaviour Aggregate
CA	Certification Authority
CHAP	Challenge – Handshake Authentication Protocol
CHDLC	Cisco HDLC
CLI	Command Line Interface
CUT	Coordinated Universal Time
DHCP	Dynamic Host Configuration Protocol
DSCP	Differentiated Services Code Point
DES	Data Encryption Standard
DiffServ	Differentiated Services
DNS	Domain Name Service
EF	Expedited Forwarding
ESP	Encapsulation Security Payload.
EWMA	Exponential Weighted Moving Average
FQDN	Fully Qualified Domain Name
HDLC	High Level Data Link Control
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineering
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IKE	Internet Key Exchange
IP	Internet Protocol
IPng	Internet Protocol next generation
IPsec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISAKMP	Internet Security Association Key Management Protocol
ISATAP	Intra-Site Automatic Tunnel Addressing Protocol

ISP	Internet Service Provider
LAN	Local Area Network
MAC	Medium Access Control
MIB	Management Information Base
MPEG	Moving Picture Experts Group
NDP	Neighbor Discovery Protocol
NMS	Network Management System
NTP	Network Time Protocol
PAP	Password Authentication Protocol
PHB	Per Hop Behaviour
PKCS	Public-Key Cryptography Standard
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PSK	Pre Shared Key
PVC	Permanent Virtual Channel
QoS	Quality of Service
RED	Random Early Drop
RFC	Request For Comment
RIO	RED with In/Out bit
RIP	Routing Internet Protocol
RIPng	Routing Internet Protocol for IPv6
RSA	Rivest Shamir Adleman algorithm
RTP	Real Time Protocol
SA	Security Association
SixOS	6WIND Operating System
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
SPI	Security Parameter Index
SSH	Secured SHell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VPN	Virtual Private Network
WAN	Wide Area Network