Foundry Security Guide

FOUNDRY NETWORKS
www.foundrynetworks.com

2100 Gold Street
P.O. Box 649100
San Jose, CA 95164-9100
Tel 408.586.1700
Fax 408.586.1900

January 2006

Contents

Chapter 1
Getting Started

Introduction

This guide describes how to configure the security features available on the following Foundry devices.

- Enterprise IronWare software releases, which apply to the following products:
  - NetIron 400/800/1500 Chassis devices with IronCore or JetCore management modules
  - BigIron 4000/8000/15000 Chassis devices with IronCore or JetCore management modules
  - FastIron II, FastIron II Plus, and FastIron III with M2 or higher management modules
  - FastIron 400/800/1500 Chassis devices with JetCore modules
  - FastIron 4802 Stackable device
- Service Provider IronWare software releases, which apply to the following products:
  - NetIron 400/800/1500 Chassis devices with IronCore or JetCore management modules
  - BigIron 4000/8000/15000 Chassis devices with IronCore or JetCore management modules
  - NetIron 4802 Stackable device
  - FastIron 4802 Stackable device

  NOTE: You cannot use this software on FastIron Chassis devices.

- Terathon devices that include the following:
  - BigIron MG8
  - NetIron 40G
  - NetIron IMR 640
- FastIron family releases that include the following devices:
  - FastIron Edge Switch
  - FastIron Edge Switch X-Series
  - FastIron SuperX release
- IronPoint-FastIron Edge Switch (IP-FES) Release 01.3.00 through 01.4.01
- ServerIron product family

For a list of enhancements in this edition, see the *Foundry Switch and Router Installation and Basic Configuration Guide*.

Audience

This manual is designed for system administrators with a working knowledge of Layer 2 and Layer 3 switching and routing.

If you are using a Foundry Layer 3 Switch, you should be familiar with the following protocols if applicable to your network – IP, RIP, OSPF, BGP4, IGMP, PIM, DVMRP, IPX, AppleTalk, FSRP, and VRRP.

Nomenclature

This guide uses the following typographical conventions to show information:

Italic	highlights the title of another publication and occasionally emphasizes a word or phrase.
Bold	highlights a CLI command.
Bold Italic	highlights a term that is being defined.
Underline	highlights a link on the Web management interface.
Capitals	highlights field names and buttons that appear in the Web management interface.

NOTE: A note emphasizes an important fact or calls your attention to a dependency.

WARNING: A warning calls your attention to a possible hazard that can cause injury or death.

CAUTION: A caution calls your attention to a possible hazard that can damage equipment.

Related Publications

The following Foundry Networks documents supplement the information in this guide.

- *Foundry Switch and Router Installation and Basic Configuration Guide* – provides configuration guidelines for Layer 2 and Layer 3 devices and installation procedures for the Foundry devices with IronCore and JetCore modules.
- *Foundry Security Guide* – provides procedures for securing management access to Foundry devices and for protecting against Denial of Service (DoS) attacks.
- *Foundry Enterprise Configuration and Management Guide* – provides configuration information for enterprise routing protocols including IP, RIP, IP multicast, OSPF, BGP4, VRRP and VRRPE.
- *Foundry NetIron Service Provider Configuration and Management Guide* – provides configuration information for IS-IS and MPLS for Foundry devices that support IS-IS and MPLS, except for the NetIron IMR 640.
- *Foundry NetIron IMR 640 Service Provider Configuration and Management Guide* – provides configuration information for IS-IS and MPLS for for the NetIron IMR 640.
- *Foundry Switch and Router Command Line Interface Reference* – provides a list and syntax information for all the Layer 2 Switch and Layer 3 Switch CLI commands.
- *Foundry Diagnostic Guide* – provides descriptions of diagnostic commands that can help you diagnose and solve issues on Layer 2 Switches and Layer 3 Switches.
- *Foundry BigIron MG8 Switch Installation and Basic Configuration Guide* – provides installation procedures for the BigIron MG8. This guide also presents the management modules available in the device.
- *Foundry NetIron 40G Switch Installation and Basic Configuration Guide* – provides installation procedures for the BigIron MG8. This guide also presents the management modules available in the device.

- *NetIron IMR 640 Installation and Basic Configuration Guide* – provides procedures for installing modules into and connecting your DC power source(s) to the NetIron IMR 640 chassis, cabling the Ethernet interface ports, and performing a basic configuration of the software.
- *Foundry Management Information Base Reference* – presents the Simple Network Management Protocol (SNMP) Management Information Base (MIB) objects that are supported in the Foundry devices.
- *Foundry IPv6 Configuration Guide* – provide configuration information for IPv6 features.
- *Foundry IronPoint Wireless LAN Configuration Guide* – presents the features for the IronPoint wireless LAN (WLAN).

To order additional copies of these manuals, do one of the following:

- Call 1.877.TURBOCALL (887.2622) in the United States or 1.408.586.1881 outside the United States.
- Send email to info@foundrynet.com.

How to Get Help

Foundry Networks technical support will ensure that the fast and easy access that you have come to expect from your Foundry Networks products will be maintained.

Web Access

- http://www.foundrynetworks.com

Email Access

Technical requests can also be sent to the following email address:

- support@foundrynet.com

Telephone Access

- 1.877.TURBOCALL (887.2622) United States
- 1.408.586.1881 Outside the United States

Warranty Coverage

Contact Foundry Networks using any of the methods listed above for information about the standard and extended warranties.

Chapter 2
Securing Access to Management Functions

This chapter explains how to secure access to management functions on a Foundry device. It contains the following sections:

- “Securing Access Methods” on page 2-2 lists the management access methods available on a Foundry device and the ways you can secure each one
- “Restricting Remote Access to Management Functions” on page 2-4 explains how to restrict access to management functions from remote sources, including Telnet, the Web management interface, and SNMP
- “Setting Passwords” on page 2-15 explains how to set passwords for Telnet access and management privilege levels
- “Setting Up Local User Accounts” on page 2-19 explains how to define user accounts to regulate who can access management functions
- “Configuring TACACS/TACACS+ Security” on page 2-23 explains how to configure SNMP read-only and read-write community strings on a Foundry device
- “Configuring TACACS/TACACS+ Security” on page 2-23 explains how to configure TACACS/TACACS+ authentication, authorization, and accounting
- “Configuring RADIUS Security” on page 2-40 explains how to configure RADIUS authentication, authorization, and accounting
- “Configuring Authentication-Method Lists” on page 2-56 explains how to set the order that authentication methods are consulted when more than one is used with an access method
- “Using the Configuration Bootguard Feature” on page 2-60

NOTE: For all Foundry devices, RADIUS Challenge is supported for 802.1x authentication but not for login authentication. Also, multiple challenges are supported for TACACS+ login authentication.

Securing Access Methods

The following table lists the management access methods available on a Foundry device, how they are secured by default, and the ways in which they can be secured.

Table 2.1: Ways to secure management access to Foundry devices

Access method	How the access method is secured by default	Ways to secure the access method	See page
Serial access to the CLI	Not secured	Establish passwords for management privilege levels	2-16
Access to the Privileged EXEC and CONFIG levels of the CLI	Not secured	Establish a password for Telnet access to the CLI	2-15
		Establish passwords for management privilege levels	2-16
		Set up local user accounts	2-19
		Configure TACACS/TACACS+ security	2-23
		Configure RADIUS security	2-40
Telnet access	Not secured	Regulate Telnet access using ACLs	2-5
		Allow Telnet access only from specific IP addresses	2-8
		Restrict Telnet access based on a client's MAC address	2-9
		Allow Telnet access only to clients connected to a specific VLAN	2-10
		Specify the maximum number of login attempts for Telnet access	2-10
		Disable Telnet access	2-12
		Establish a password for Telnet access	2-15
		Establish passwords for privilege levels of the CLI	2-16
		Set up local user accounts	2-19
		Configure TACACS/TACACS+ security	2-23
		Configure RADIUS security	2-40

Table 2.1: Ways to secure management access to Foundry devices (Continued)

Access method	How the access method is secured by default	Ways to secure the access method	See page
Secure Shell (SSH) access	Not configured	Configure SSH	3-1
		Regulate SSH access using ACLs	2-5
		Allow SSH access only from specific IP addresses	2-8
		Restrict SSH access based on a client's MAC address	2-9
		Establish passwords for privilege levels of the CLI	2-16
		Set up local user accounts	2-19
		Configure TACACS/TACACS+ security	2-23
		Configure RADIUS security	2-40
Web management access	SNMP read or read-write community strings	Regulate Web management access using ACLs	2-6
		Allow Web management access only from specific IP addresses	2-8
		Allow Web management access only to clients connected to a specific VLAN	2-10
		Disable Web management access	2-12
		Configure SSL security for the Web management interface	2-21
		Set up local user accounts	2-19
		Establish SNMP read or read-write community strings for SNMP versions 1 and 2	14-1
		Establishing user groups for SNMP version 3	14-7
		Configure TACACS/TACACS+ security	2-23
		Configure RADIUS security	2-40

Table 2.1: Ways to secure management access to Foundry devices (Continued)

Access method	How the access method is secured by default	Ways to secure the access method	See page
SNMP (IronView Network Manager) access	SNMP read or read-write community strings and the password to the Super User privilege level **Note**: SNMP read or read-write community strings are always required for SNMP access to the device.	Regulate SNMP access using ACLs	2-6
		Allow SNMP access only from specific IP addresses	2-9
		Disable SNMP access	2-14
		Allow SNMP access only to clients connected to a specific VLAN	2-11
		Establish passwords to management levels of the CLI	2-16
		Set up local user accounts	2-19
		Establish SNMP read or read-write community strings	2-23
TFTP access	Not secured	Allow TFTP access only to clients connected to a specific VLAN	2-11

Restricting Remote Access to Management Functions

You can restrict access to management functions from remote sources, including Telnet, the Web management interface, and SNMP. The following methods for restricting remote access are supported:

- Using ACLs to restrict Telnet, Web management interface, or SNMP access
- Allowing remote access only from specific IP addresses
- Allowing remote access only to clients connected to a specific VLAN
- Specifically disabling Telnet, Web management interface, or SNMP access to the device

The following sections describe how to restrict remote access to a Foundry device using these methods.

Using ACLs to Restrict Remote Access

You can use standard ACLs to control the following access methods to management functions on a Foundry device:

- Telnet access
- SSH access
- Web management access
- SNMP access

To configure access control for these management access methods:

1. Configure an ACL with the IP addresses you want to allow to access the device
2. Configure a Telnet access group, SSH access group, web access group, and SNMP community strings. Each of these configuration items accepts an ACL as a parameter. The ACL contains entries that identify the IP addresses that can use the access method.

The following sections present examples of how to secure management access using ACLs. See the “IP Access Control Lists (ACLs)” chapter in the *Foundry Enterprise Configuration and Management Guide* for more information on configuring ACLs.

NOTE: In releases prior to 07.7.00, ACL filtering for remote management access was done in software (that is, by the CPU). Starting with release 07.7.00, you can configure JetCore devices to perform the filtering in hardware. See "Hardware Filtering for Remote Management Access (JetCore Devices Running Release 07.7.00 and Higher)" on page 2-7.

Using an ACL to Restrict Telnet Access

To configure an ACL that restricts Telnet access to the device, enter commands such as the following:

```
BigIron(config)# access-list 10 deny host 209.157.22.32 log
BigIron(config)# access-list 10 deny 209.157.23.0 0.0.0.255 log
BigIron(config)# access-list 10 deny 209.157.24.0 0.0.0.255 log
BigIron(config)# access-list 10 deny 209.157.25.0/24 log
BigIron(config)# access-list 10 permit any
BigIron(config)# telnet access-group 10
BigIron(config)# write memory
```

Syntax: telnet access-group <num>

The <num> parameter specifies the number of a standard ACL and must be from 1 – 99.

The commands above configure ACL 10, then apply the ACL as the access list for Telnet access. The device allows Telnet access to all IP addresses except those listed in ACL 10.

To configure a more restrictive ACL, create permit entries and omit the **permit any** entry at the end of the ACL. For example:

```
BigIron(config)# access-list 10 permit host 209.157.22.32
BigIron(config)# access-list 10 permit 209.157.23.0 0.0.0.255
BigIron(config)# access-list 10 permit 209.157.24.0 0.0.0.255
BigIron(config)# access-list 10 permit 209.157.25.0/24
BigIron(config)# telnet access-group 10
BigIron(config)# write memory
```

The ACL in this example permits Telnet access only to the IP addresses in the **permit** entries and denies Telnet access from all other IP addresses.

Using an ACL to Restrict SSH Access

To configure an ACL that restricts SSH access to the device, enter commands such as the following:

```
BigIron(config)# access-list 12 deny host 209.157.22.98 log
BigIron(config)# access-list 12 deny 209.157.23.0 0.0.0.255 log
BigIron(config)# access-list 12 deny 209.157.24.0/24 log
BigIron(config)# access-list 12 permit any
BigIron(config)# ssh access-group 12
BigIron(config)# write memory
```

Syntax: ssh access-group <num>

The <num> parameter specifies the number of a standard ACL and must be from 1 – 99.

These commands configure ACL 12, then apply the ACL as the access list for SSH access. The device denies SSH access from the IP addresses listed in ACL 12 and permits SSH access from all other IP addresses. Without the last ACL entry for permitting all packets, this ACL would deny SSH access from all IP addresses.

NOTE: In this example, the command **ssh access-group 10** could have been used to apply the ACL configured in the example for Telnet access. You can use the same ACL multiple times.

Using an ACL to Restrict Web Management Access

To configure an ACL that restricts Web management access to the device, enter commands such as the following:

```
BigIron(config)# access-list 12 deny host 209.157.22.98 log
BigIron(config)# access-list 12 deny 209.157.23.0 0.0.0.255 log
BigIron(config)# access-list 12 deny 209.157.24.0/24 log
BigIron(config)# access-list 12 permit any
BigIron(config)# web access-group 12
BigIron(config)# write memory
```

Syntax: web access-group <num>

The <num> parameter specifies the number of a standard ACL and must be from 1 – 99.

These commands configure ACL 12, then apply the ACL as the access list for Web management access. The device denies Web management access from the IP addresses listed in ACL 12 and permits Web management access from all other IP addresses. Without the last ACL entry for permitting all packets, this ACL would deny Web management access from all IP addresses.

Using ACLs to Restrict SNMP Access

To restrict SNMP access to the device using ACLs, enter commands such as the following:

NOTE: The syntax for using ACLs for SNMP access is different from the syntax for controlling Telnet, SSH, and Web management access using ACLs.

```
BigIron(config)# access-list 25 deny host 209.157.22.98 log
BigIron(config)# access-list 25 deny 209.157.23.0 0.0.0.255 log
BigIron(config)# access-list 25 deny 209.157.24.0 0.0.0.255 log
BigIron(config)# access-list 25 permit any
BigIron(config)# access-list 30 deny 209.157.25.0 0.0.0.255 log
BigIron(config)# access-list 30 deny 209.157.26.0/24 log
BigIron(config)# access-list 30 permit any
BigIron(config)# snmp-server community public ro 25
BigIron(config)# snmp-server community private rw 30
BigIron(config)# write memory
```

Syntax: snmp-server community <string> ro | rw <num>

The <string> parameter specifies the SNMP community string the user must enter to gain SNMP access.

The **ro** parameter indicates that the community string is for read-only (“get”) access. The **rw** parameter indicates the community string is for read-write (“set”) access.

The <num> parameter specifies the number of a standard ACL and must be from 1 – 99.

These commands configure ACLs 25 and 30, then apply the ACLs to community strings.

ACL 25 is used to control read-only access using the “public” community string. ACL 30 is used to control read-write access using the “private” community string.

NOTE: When **snmp-server community** is configured, all incoming SNMP packets are validated first by their community strings and then by their bound ACLs. Before software release 07.7.00, packets are denied if filters are not configured for an ACL. Beginning with software release 07.7.00, packets are permitted if no filters are configured for an ACL.

Hardware Filtering for Remote Management Access (JetCore Devices Running Release 07.7.00 and Higher)

In releases prior to 07.7.00, ACL filtering for remote management access was done in software (that is, by the CPU). Starting with release 07.7.00, you can configure JetCore devices to perform the filtering in hardware.

Configuring Hardware-Based Remote Access Filtering on Layer 2 Switches

The following is an example of configuring a JetCore Layer 2 Switch to perform hardware filtering for Telnet access.

```
FastIron(config)# vlan 3 by port
FastIron(config-vlan-3)# untagged ethe 3/1 to 3/5
FastIron(config-vlan-3)# exit

FastIron(config)# ip address 10.10.11.1 255.255.255.0
FastIron(config)# telnet access-group 10 vlan 3
FastIron(config)# access-list 10 deny host 10.10.11.2
FastIron(config)# access-list 10 deny host 10.10.11.3
FastIron(config)# access-list 10 deny host 10.10.11.4
FastIron(config)# access-list 10 deny host 10.10.11.5
```

In this example, a Layer 2 VLAN is configured as a remote-access management VLAN. The IP address configured for the Foundry device is also used for the management IP address of the Layer 2 VLAN. Hardware filtering of Telnet traffic is performed for all the ports in the management VLAN.

You can display information about the hardware filtering with the **show cam l4** command. For example:

```
FastIron# show cam l4 3/2
Sl Index       Src IP_Addr  SPort      Dest IP_Addr  DPort Prot Age    Out Port
 3 24577      10.10.11.2/32   Any      10.10.11.1/24    23  TCP dis     Discard
 3 24579      10.10.11.3/32   Any      10.10.11.1/24    23  TCP dis     Discard
 3 24581      10.10.11.4/32   Any      10.10.11.1/24    23  TCP dis     Discard
 3 24583      10.10.11.5/32   Any      10.10.11.1/24    23  TCP dis     Discard
```

The source IP address of the filter entry is taken from the standard ACL, and the destination IP address of the filter entry is taken from the management IP address of the Layer 2 Switch.

Configuring Hardware-Based Remote Access Filtering on Layer 3 Switches

The following is an example of configuring a JetCore Layer 3 Switch to perform hardware filtering for Telnet access.

```
BigIron(config)# vlan 3 by port
BigIron(config-vlan-3)# untagged ethe 3/1 to 3/5
BigIron(config-vlan-3)# router-interface ve 3
BigIron(config-vlan-3)# exit

BigIron(config)# interface ve 3
BigIron(config-ve-1)# ip address 10.10.11.1 255.255.255.0
BigIron(config-ve-1)# exit

BigIron(config)# access-list 10 permit host 10.10.11.254
BigIron(config)# access-list 10 permit host 192.168.2.254
BigIron(config)# access-list 10 permit host 192.168.12.254
BigIron(config)# access-list 10 permit host 192.64.22.254
BigIron(config)# access-list 10 deny any

BigIron(config)# telnet access-group 10 vlan 3
BigIron(config)# ssh access-group 10 vlan 3
BigIron(config)# web access-group 10 vlan 3
BigIron(config)# snmp-server community private rw 10 vlan 3
```

In this example, a Layer 3 VLAN is configured as a remote-access management VLAN and a router interface. The IP address specified for the router interface becomes the management IP address of the VLAN.

When you make changes to the ACL configuration and/or make changes to the management VLAN, you must enter the following command after making the configuration changes:

```
BigIron(config)# remote-management rebind
```

Syntax: remote-management rebind

The **show cam l4** command displays the following information about the hardware filtering in this configuration:

```
BigIron# show cam l4 3/1
Sl Index       Src IP_Addr  SPort      Dest IP_Addr  DPort Prot Age    Out Port
 3 40960   192.64.22.254/32   Any      10.10.11.1/24    23  TCP dis   Use L2/L3
 3 40962  192.168.12.254/32   Any      10.10.11.1/24    23  TCP dis   Use L2/L3
 3 40964   192.168.2.254/32   Any      10.10.11.1/24    23  TCP dis   Use L2/L3
 3 40966    10.10.11.254/32   Any      10.10.11.1/24    23  TCP dis   Use L2/L3
 3 40968               Any     Any      10.10.11.1/24     23   TCP dis     Discard
```

The IP address in standard ACL 10 is the source IP address of the filter entry, and the IP address of the router interface is the destination IP address of the filter entry.

Restricting Remote Access to the Device to Specific IP Addresses

By default, a Foundry device does not control remote management access based on the IP address of the managing device. You can restrict remote management access to a single IP address for the following access methods:

- Telnet access
- Web management access
- SNMP access

In addition, if you want to restrict all three access methods to the same IP address, you can do so using a single command.

The following examples show the CLI commands for restricting remote access. You can specify only one IP address with each command. However, you can enter each command ten times to specify up to ten IP addresses.

NOTE: You cannot restrict remote management access using the Web management interface.

Restricting Telnet Access to a Specific IP Address

To allow Telnet access to the Foundry device only to the host with IP address 209.157.22.39, enter the following command:

```
BigIron(config)# telnet-client 209.157.22.39
```

Syntax: [no] telnet-client <ip-addr>

Restricting SSH Access to a Specific IP Address

To allow SSH access to the Foundry device only to the host with IP address 209.157.22.39, enter the following command:

```
BigIron(config)# ip ssh client 209.157.22.39
```

Syntax: [no] ip ssh client <ip-addr>

Restricting Web Management Access to a Specific IP Address

To allow Web management access to the Foundry device only to the host with IP address 209.157.22.26, enter the following command:

```
BigIron(config)# web-client 209.157.22.26
```

Syntax: [no] web-client <ip-addr>

Restricting SNMP Access to a Specific IP Address

To allow SNMP access (which includes IronView Network Manager) to the Foundry device only to the host with IP address 209.157.22.14, enter the following command:

```
BigIron(config)# snmp-client 209.157.22.14
```

Syntax: [no] snmp-client <ip-addr>

Restricting All Remote Management Access to a Specific IP Address

To allow Telnet, Web, and SNMP management access to the Foundry device only to the host with IP address 209.157.22.69, you can enter three separate commands (one for each access type) or you can enter the following command:

```
BigIron(config)# all-client 209.157.22.69
```

Syntax: [no] all-client <ip-addr>

Restricting Telnet and SSH Access Based on a Client's MAC Address

Starting in release 02.0.00 for the FastIron SuperX and release 07.8.00, you can restrict remote management access to the Foundry device based on the MAC address of a connecting client. This feature applies to Telnet and SSH access to the device.

For the FastIron SuperX, previous releases allow you to restrict Telnet (as well as Web Management interface and SNMP) access based on a client's IP address. For example, the following command allows Telnet access to the Foundry device only to the host with IP address 209.157.22.39:

```
FastIron SuperX Switch(config)# telnet-client 209.157.22.39
```

Syntax: [no] telnet-client <ip-addr>

This functionality is retained in release 02.0.00 for the FastIron SuperX. In addition, you can now optionally limit Telnet or SSH access to clients with a specified MAC address.

For example, the following command allows Telnet access to the Foundry device only to the host with IP address 209.157.22.39 ***and*** MAC address 0007.e90f.e9a0:

```
BigIron(config)# telnet-client 209.157.22.39 0007.e90f.e9a0
```

Syntax: [no] telnet-client <ip-addr> <mac-addr>

The following command allows Telnet access to the Foundry device to a host with any IP address and MAC address 0007.e90f.e9a0:

```
BigIron(config)# telnet-client any 0007.e90f.e9a0
```

Syntax: [no] telnet-client any <mac-addr>

In release 02.0.00 for the FastIron SuperX, to allow SSH access to the Foundry device only to the host with IP address 209.157.22.39, enter the following command:

```
FastIron SuperX Switch(config)# ip ssh client 209.157.22.39
```

Syntax: [no] ip ssh client <ip-addr>

To allow SSH access to the Foundry device only to the host with IP address 209.157.22.39 ***and*** MAC address 0007.e90f.e9a0, enter the following command:

```
BigIron(config)# ip ssh client 209.157.22.39 0007.e90f.e9a0
```

Syntax: [no] ip ssh client <ip-addr> <mac-addr>

To allow SSH access to the Foundry device to a host with any IP address and MAC address 0007.e90f.e9a0, enter the following command:

```
BigIron(config)# ip ssh client any 0007.e90f.e9a0
```

Syntax: [no] ip ssh client any <mac-addr>

Specifying the Maximum Number of Login Attempts for Telnet Access

NOTE: This feature is also available in release 03.3.00 for the FES and release 02.0.00 for the FastIron SuperX.

If you are connecting to the Foundry device using Telnet, the device prompts you for a username and password. By default, you have up to 4 chances to enter a correct username and password. If you do not enter a correct username or password after 4 attempts, the Foundry device disconnects the Telnet session.

You can specify the number of attempts a Telnet user has to enter a correct username and password before the device disconnects the Telnet session. For example, to allow a Telnet user up to 5 chances to enter a correct username and password, enter the following command:

```
BigIron(config)# telnet login-retries 5
```

Syntax: [no] telnet login-retries <number>

You can specify from 0 – 5 attempts. The default is 4 attempts.

Restricting Remote Access to the Device to Specific VLAN IDs

You can restrict management access to a Foundry device to ports within a specific port-based VLAN. VLAN-based access control applies to the following access methods:

- Telnet access
- Web management access
- SNMP access
- TFTP access

By default, access is allowed for all the methods listed above on all ports. Once you configure security for a given access method based on VLAN ID, access to the device using that method is restricted to only the ports within the specified VLAN.

VLAN-based access control works in conjunction with other access control methods. For example, suppose you configure an ACL to permit Telnet access only to specific client IP addresses, and you also configure VLAN-based access control for Telnet access. In this case, the only Telnet clients that can access the device are clients that have one of the IP addresses permitted by the ACL ***and*** are connected to a port that is in a permitted VLAN. Clients who have a permitted IP address but are connected to a port in a VLAN that is not permitted still cannot access the device through Telnet.

Restricting Telnet Access to a Specific VLAN

To allow Telnet access only to clients in a specific VLAN, enter a command such as the following:

```
BigIron(config)# telnet server enable vlan 10
```

The command in this example configures the device to allow Telnet management access only to clients connected to ports within port-based VLAN 10. Clients connected to ports that are not in VLAN 10 are denied management access.

Syntax: [no] telnet server enable vlan <vlan-id>

Restricting Web Management Access to a Specific VLAN

To allow Web management access only to clients in a specific VLAN, enter a command such as the following:

```
BigIron(config)# web-management enable vlan 10
```

The command in this example configures the device to allow Web management access only to clients connected to ports within port-based VLAN 10. Clients connected to ports that are not in VLAN 10 are denied management access.

Syntax: [no] web-management enable vlan <vlan-id>

Restricting SNMP Access to a Specific VLAN

To allow SNMP access only to clients in a specific VLAN, enter a command such as the following:

```
BigIron(config)# snmp-server enable vlan 40
```

The command in this example configures the device to allow SNMP access only to clients connected to ports within port-based VLAN 40. Clients connected to ports that are not in VLAN 40 are denied access.

Syntax: [no] snmp-server enable vlan <vlan-id>

Restricting TFTP Access to a Specific VLAN

To allow TFTP access only to clients in a specific VLAN, enter a command such as the following:

```
BigIron(config)# tftp client enable vlan 40
```

The command in this example configures the device to allow TFTP access only to clients connected to ports within port-based VLAN 40. Clients connected to ports that are not in VLAN 40 are denied access.

Syntax: [no] tftp client enable vlan <vlan-id>

Designated VLAN for Telnet Management Sessions to a Layer 2 Switch

By default, the management IP address you configure on a Layer 2 Switch applies globally to all the ports on the device. This is true even if you divide the device's ports into multiple port-based VLANs.

If you want to restrict the IP management address to a specific port-based VLAN, you can make that VLAN the designated management VLAN for the device. When you configure a VLAN to be the designated management VLAN, the management IP address you configure on the device is associated only with the ports in the designated VLAN. To establish a Telnet management session with the device, a user must access the device through one of the ports in the designated VLAN.

You also can configure up to five default gateways for the designated VLAN, and associate a metric with each one. The software uses the gateway with the lowest metric. The other gateways reside in the configuration but are not used. You can use one of the other gateways by modifying the configuration so that the gateway you want to use has the lowest metric.

If more than one gateway has the lowest metric, the software uses the gateway that appears first in the running-config.

NOTE: If you have already configured a default gateway globally and you do not configure a gateway in the VLAN, the software uses the globally configured gateway and gives the gateway a metric value of 1.

To configure a designated management VLAN, enter commands such as the following:

```
FastIron(config)# vlan 10 by port
FastIron(config-vlan-10)# untag ethernet 1/1 to 1/4
FastIron(config-vlan-10)# management-vlan
FastIron(config-vlan-10)# default-gateway 10.10.10.1 1
FastIron(config-vlan-10)# default-gateway 20.20.20.1 2
```

These commands configure port-based VLAN 10 to consist of ports 1/1 – 1/4 and to be the designated management VLAN. The last two commands configure default gateways for the VLAN. Since the 10.10.10.1 gateway has a lower metric, the software uses this gateway. The other gateway remains in the configuration but is not used. You can use the other one by changing the metrics so that the 20.20.20.1 gateway has the lower metric.

Syntax: [no] management-vlan

Syntax: [no] default-gateway <ip-addr> <metric>

The <ip-addr> parameters specify the IP address of the gateway router.

The <metric> parameter specifies the metric (cost) of the gateway. You can specify a value from 1 – 5. There is no default. The software uses the gateway with the lowest metric.

Disabling Specific Access Methods

You can specifically disable the following access methods:

- Telnet access
- Web management access
- SNMP access

NOTE: If you disable Telnet access, you will not be able to access the CLI except through a serial connection to the management module. If you disable SNMP access, you will not be able to use IronView Network Manager or third-party SNMP management applications.

NOTE: In software releases 07.7.00 and later, you can disable access to the Management IP address through the device's Content Addressable Memory (CAM). See "Disabling an Interface's Access to Management Functions" on page 2-14.

Disabling Telnet Access

Telnet access is enabled by default. You can use a Telnet client to access the CLI on the device over the network. If you do not plan to use the CLI over the network and want to disable Telnet access to prevent others from establishing CLI sessions with the device, enter the following command:

```
BigIron(config)# no telnet-server
```

To re-enable Telnet operation, enter the following command:

```
BigIron(config)# telnet-server
```

Syntax: [no] telnet-server

Disabling Web Management Access

If you want to prevent access to the device through the Web management interface, you can disable the Web management interface.

NOTE: As soon as you make this change, the device stops responding to Web management sessions. If you make this change using your Web browser, your browser can contact the device, but the device will not reply once the change takes place.

USING THE CLI

To disable the Web management interface, enter the following command:

```
BigIron(config)# no web-management
```

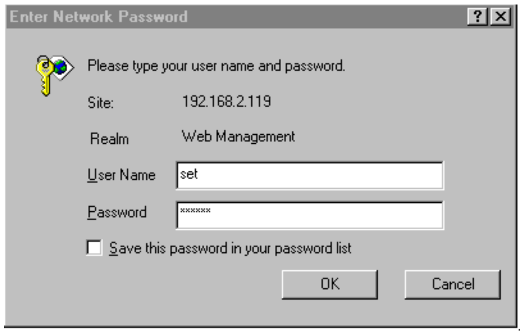
To re-enable the Web management interface, enter the following command:

```
BigIron(config)# web-management
```

Syntax: [no] web-management

USING THE WEB MANAGEMENT INTERFACE

1. Log on to the device using a valid user name and password for read-write access.



Please type your user name and password.

Site: 192.168.2.119

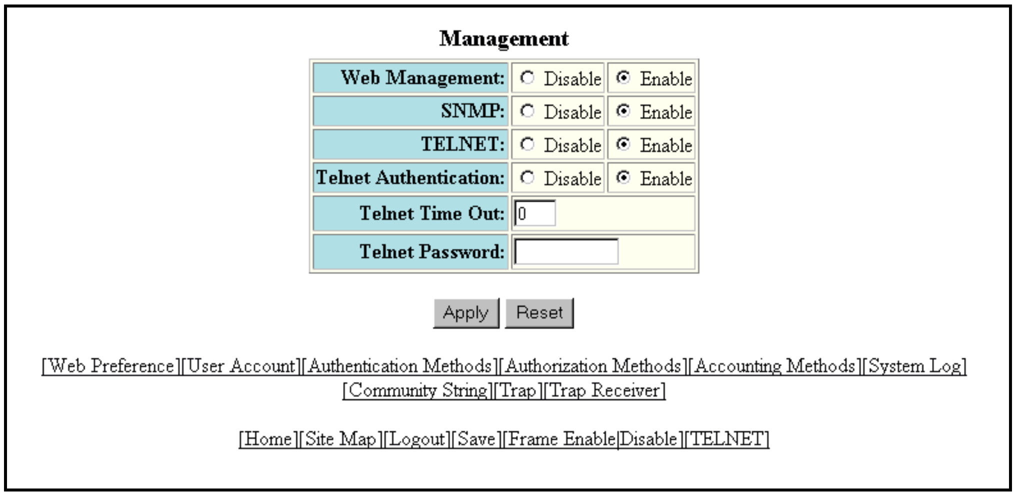
Realm Web Management

User Name set

Password ******

Save this password in your password list

2. Select the Management link from the System configuration panel to display the Management configuration panel.



Management

Web Management:	Disable	[✓ Enable]
SNMP:	Disable	[✓ Enable]
TELNET:	Disable	[✓ Enable]
Telnet Authentication:	Disable	[✓ Enable]
Telnet Time Out:	0	
Telnet Password:		

Apply
Reset
[Web Preference][User Account][Authentication Methods][Authorization Methods][Accounting Methods][System Log]
[Community String][Trap][Trap Receiver]
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

3. Click Disable next to Web Management.
4. Click the Apply button to save the change to the device's running-config file.
5. Select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

Disabling Web Management Access by HP ProCurve Manager

By default, TCP ports 80 and 280 are enabled on the Foundry device. TCP port 80 (HTTP) allows access to the device's Web management interface. TCP port 280 allows access to the device by HP ProCurve Manager.

The **no web-management** command disables both TCP ports. However, if you want to disable only port 280 and leave port 80 enabled, use the **hp-top-tools** option with the command. Here is an example.

```
BigIron(config)# no web-management hp-top-tools
```

Syntax: [no] web-management [allow-no-password | enable [vlan <vlan-id>] | front-panel | hp-top-tools | list-menu]

The **hp-top-tools** parameter disables TCP port 280. For information about the other parameters, see the *Foundry Switch and Router Command Line Interface Reference*

Disabling SNMP Access

SNMP is enabled by default on all Foundry devices. SNMP is required if you want to manage a Foundry device using IronView Network Manager.

To disable SNMP, use one of the following methods.

USING THE CLI

To disable SNMP management of the device:

```
BigIron(config)# snmp disable
```

To later re-enable SNMP management of the device:

```
BigIron(config)# no snmp disable
```

Syntax: [no] snmp disable

USING THE WEB MANAGEMENT INTERFACE

1. Log on to the device using a valid user name and password for read-write access. The System configuration dialog is displayed.
2. Select the Management link from the System configuration panel to display the Management configuration panel.
3. Click Disable next to SNMP.
4. Click the Apply button to save the change to the device's running-config file.
5. Select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

Disabling an Interface's Access to Management Functions

Beginning In software release 07.7.00, you can protect the CPU from remote access to management functions such as:

- Telnet
- SSH
- Web Management Interface
- SNMP
- TFTP
- RADIUS
- TACACS
- TACACS+

To enable this feature, disable access to the Management IP address through the device's Content Addressable Memory (CAM). The following shows an example configuration.

NOTE: This feature does not affect Layer 3 routing functions.

```
BigIron(config)# int e 3/10
BigIron(config-if-e1000-3/10)# ip address 10.10.10.1 255.255.255.0
BigIron(config-if-e1000-3/10)# exit

BigIron(config)# int e 3/11
BigIron(config-if-e1000-3/11)# ip address 11.11.11.1 255.255.255.0
BigIron(config-if-e1000-3/11)# management-ip-disable
BigIron(config-if-e1000-3/11)# exit

BigIron(config)# int e 3/12
BigIron(config-if-e1000-3/12)# ip address 12.12.12.1 255.255.255.0
BigIron(config-if-e1000-3/12)# management-ip-disable
BigIron(config-if-e1000-3/12)# exit
```

```
BigIron(config)# int e 3/13
BigIron(config-if-e1000-3/13)# ip address 13.13.13.1 255.255.255.0
BigIron(config-if-e1000-3/13)# management-ip-disable
BigIron(config-if-e1000-3/13)# exit
```

Syntax: [no] ip address <ip-addr> <ip-mask>

where <ip-addr> and <ip-mask> are the destination IP address and subnet mask.

Syntax: [no] management-ip-disable

Use the **no** form of the command to re-enable access to the Management IP address.

Viewing Information about Disabled Management IP Addresses

Use the **show cam l4** command to display information about CAM entries for disabled management IP addresses.

Setting Passwords

Passwords can be used to secure the following access methods:

- Telnet access can be secured by setting a Telnet password. See “Setting a Telnet Password” on page 2-15.
- Access to the Privileged EXEC and CONFIG levels of the CLI can be secured by setting passwords for management privilege levels. See “Setting Passwords for Management Privilege Levels” on page 2-16.

This section also provides procedures for enhancing management privilege levels, recovering from a lost password, and disabling password encryption.

NOTE: You also can configure up to 16 user accounts consisting of a user name and password, and assign each user account a management privilege level. See “Setting Up Local User Accounts” on page 2-19.

Setting a Telnet Password

By default, the device does not require a user name or password when you log in to the CLI using Telnet. You can assign a password for Telnet access using one of the following methods.

USING THE CLI

To set the password “letmein” for Telnet access to the CLI, enter the following command at the global CONFIG level:

```
BigIron(config)# enable telnet password letmein
```

Syntax: [no] enable telnet password <string>

USING THE WEB MANAGEMENT INTERFACE

1. Log on to the device using a valid user name and password for read-write access. The System configuration panel is displayed.
2. Select the Management link from the System configuration panel to display the Management configuration panel.
3. Enter the password in the Telnet Password field.
4. Click the Apply button to save the change to the device’s running-config file.
5. Select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device’s flash memory.

Suppressing Telnet Connection Rejection Messages

By default, if a Foundry device denies Telnet management access to the device, the software sends a message to the denied Telnet client. You can optionally suppress the rejection message. When you enable the option, a denied Telnet client does not receive a message from the Foundry device. Instead, the denied client simply does not gain access.

To suppress the connection rejection message, use the following CLI method.

USING THE CLI

To suppress the connection rejection message sent by the device to a denied Telnet client, enter the following command at the global CONFIG level of the CLI:

```
BigIron(config)# telnet server suppress-reject-message
```

Syntax: [no] telnet server suppress-reject-message

USING THE WEB MANAGEMENT INTERFACE

You cannot configure this option using the Web management interface.

Setting Passwords for Management Privilege Levels

You can set one password for each of the following management privilege levels:

- Super User level – Allows complete read-and-write access to the system. This is generally for system administrators and is the only management privilege level that allows you to configure passwords.
- Port Configuration level – Allows read-and-write access for specific ports but not for global (system-wide) parameters.
- Read Only level – Allows access to the Privileged EXEC mode and CONFIG mode of the CLI but only with read access.

You can assign a password to each management privilege level. You also can configure up to 16 user accounts consisting of a user name and password, and assign each user account to one of the three privilege levels. See “Setting Up Local User Accounts” on page 2-19.

NOTE: You must use the CLI to assign a password for management privilege levels. You cannot assign a password using the Web management interface.

If you configure user accounts in addition to privilege level passwords, the device will validate a user’s access attempt using one or both methods (local user account or privilege level password), depending on the order you specify in the authentication-method lists. See “Configuring Authentication-Method Lists” on page 2-56.

USING THE CLI

To set passwords for management privilege levels:

1. At the opening CLI prompt, enter the following command to change to the Privileged level of the EXEC mode:

   ```
   BigIron> enable
   BigIron#
   ```

2. Access the CONFIG level of the CLI by entering the following command:

   ```
   BigIron# configure terminal
   BigIron(config)#
   ```

3. Enter the following command to set the Super User level password:

   ```
   BigIron(config)# enable super-user-password <text>
   ```

   NOTE: You must set the Super User level password before you can set other types of passwords. The Super User level password can be an alphanumeric string, but cannot begin with a number.

4. Enter the following commands to set the Port Configuration level and Read Only level passwords:

```
    BigIron(config)# enable port-config-password <text>
    BigIron(config)# enable read-only-password <text>
```

NOTE: If you forget your Super User level password, see "Recovering from a Lost Password" on page 2-18.

Augmenting Management Privilege Levels

Each management privilege level provides access to specific areas of the CLI by default:

- Super User level provides access to all commands and displays.
- Port Configuration level gives access to:
    - The User EXEC and Privileged EXEC levels
    - The port-specific parts of the CONFIG level
    - All interface configuration levels
- Read Only level gives access to:
    - The User EXEC and Privileged EXEC levels

You can grant additional access to a privilege level on an individual command basis. To grant the additional access, you specify the privilege level you are enhancing, the CLI level that contains the command, and the individual command.

NOTE: This feature applies only to management privilege levels on the CLI. You cannot augment management access levels for the Web management interface.

To enhance the Port Configuration privilege level so users also can enter IP commands at the global CONFIG level:

```
BigIron(config)# privilege configure level 4 ip
```

In this command, **configure** specifies that the enhanced access is for a command at the global CONFIG level of the CLI. The **level 4** parameter indicates that the enhanced access is for management privilege level 4 (Port Configuration). All users with Port Configuration privileges will have the enhanced access. The **ip** parameter indicates that the enhanced access is for the IP commands. Users who log in with valid Port Configuration level user names and passwords can enter commands that begin with "ip" at the global CONFIG level.

Syntax: [no] privilege <cli-level> level <privilege-level> <command-string>

The <cli-level> parameter specifies the CLI level and can be one of the following values:

- **exec** – EXEC level; for example, BigIron> or BigIron#
- **configure** – CONFIG level; for example, BigIron`(config)#`
- **interface** – Interface level; for example, BigIron`(config-if-6)#`
- **virtual-interface** – Virtual-interface level; for example, BigIron`(config-vif-6)#`
- **rip-router** – RIP router level; for example, BigIron`(config-rip-router)#`
- **ospf-router** – OSPF router level; for example, BigIron`(config-ospf-router)#`
- **dvmrp-router** – DVMRP router level; for example, BigIron`(config-dvmrp-router)#`
- **pim-router** – PIM router level; for example, BigIron`(config-pim-router)#`
- **bgp-router** – BGP4 router level; for example, BigIron`(config-bgp-router)#`
- **port-vlan** – Port-based VLAN level; for example, BigIron`(config-vlan)#`
- **protocol-vlan** – Protocol-based VLAN level

The <privilege-level> indicates the number of the management privilege level you are augmenting. You can specify one of the following:

- **0** – Super User level (full read-write access)
- **4** – Port Configuration level
- **5** – Read Only level

The <command-string> parameter specifies the command you are allowing users with the specified privilege level to enter. To display a list of the commands at a CLI level, enter “?” at that level's command prompt.

Recovering from a Lost Password

Recovery from a lost password requires direct access to the serial port and a system reset.

NOTE: You can perform this procedure only from the CLI.

To recover from a lost password:

1. Start a CLI session over the serial interface to the device.
2. Reboot the device.
3. At the initial boot prompt at system startup, enter **b** to enter the boot monitor mode.
4. Enter **no password** at the prompt. (You cannot abbreviate this command.) This command will cause the device to bypass the system password check.
5. Enter **boot system flash primary** at the prompt.
6. After the console prompt reappears, assign a new password.

Displaying the SNMP Community String

If you want to display the SNMP community string, enter the following commands:

```
BigIron(config)# enable password-display
BigIron(config)# show snmp server
```

The **enable password-display** command enables display of the community string, but only in the output of the **show snmp server** command. Display of the string is still encrypted in the startup-config file and running-config. Enter the command at the global CONFIG level of the CLI.

Disabling Password Encryption

When you configure a password, then save the configuration to the Foundry device’s flash memory, the password is also saved to flash as part of the configuration file. By default, the passwords are encrypted so that the passwords cannot be observed by another user who displays the configuration file. Even if someone observes the file while it is being transmitted over TFTP, the password is encrypted.

NOTE: You cannot disable password encryption using the Web management interface.

If you want to remove the password encryption, you can disable encryption by entering the following command:

```
BigIron(config)# no service password-encryption
```

Syntax: [no] service password-encryption

Specifying a Minimum Password Length

By default, the Foundry device imposes no minimum length on the Line (Telnet), Enable, or Local passwords. You can configure the device to require that Line, Enable, and Local passwords be at least a specified length.

For example, to specify that the Line, Enable, and Local passwords be at least 8 characters, enter the following command:

```
BigIron(config)# enable password-min-length 8
```

Syntax: enable password-min-length <number-of-characters>

The <number-of-characters> can be from 1 – 48.

Setting Up Local User Accounts

You can define up to 16 local user accounts on a Foundry device. User accounts regulate who can access the management functions in the CLI using the following methods:

- Telnet access
- Web management access
- SNMP access

NOTE: Local user accounts are not supported on the FastIron Workgroup Layer 2 Switch or the non-octal NetIron.

Local user accounts provide greater flexibility for controlling management access to Foundry devices than do management privilege level passwords and SNMP community strings of SNMP versions 1 and 2. You can continue to use the privilege level passwords and the SNMP community strings as additional means of access authentication. Alternatively, you can choose not to use local user accounts and instead continue to use only the privilege level passwords and SNMP community strings. Local user accounts are backward-compatible with configuration files that contain privilege level passwords. See "Setting Passwords for Management Privilege Levels" on page 2-16.

If you configure local user accounts, you also need to configure an authentication-method list for Telnet access, Web management access, and SNMP access. See "Configuring Authentication-Method Lists" on page 2-56.

For each local user account, you specify a user name. You also can specify the following parameters:

- A password
- A management privilege level, which can be one of the following:
  - Super User level – Allows complete read-and-write access to the system. This is generally for system administrators and is the only privilege level that allows you to configure passwords. This is the default.
  - Port Configuration level – Allows read-and-write access for specific ports but not for global (system-wide) parameters.
  - Read Only level – Allows access to the Privileged EXEC mode and CONFIG mode but only with read access.

Configuring a Local User Account

To configure a local user account, use one of the following methods.

USING THE CLI

To configure a local user account, enter a command such as the following at the global CONFIG level of the CLI.

```
BigIron(config)# username wonka password willy
```

This command adds a local user account with the user name "wonka" and the password "willy". This account has the Super User privilege level; this user has full access to all configuration and display features.

NOTE: If you configure local user accounts, you must grant Super User level access to at least one account before you add accounts with other privilege levels. You need the Super User account to make further administrative changes.

```
BigIron(config)# username waldo privilege 5 password whereis
```

This command adds a user account for user name "waldo", password "whereis", with the Read Only privilege level. Waldo can look for information but cannot make configuration changes.

Syntax: [no] username <user-string> privilege <privilege-level> password | nopassword <password-string>

You can enter up to 255 characters for <user-string>.

The **privilege** parameter specifies the privilege level for the account. You can specify one of the following:

- **0** – Super User level (full read-write access)
- **4** – Port Configuration level
- **5** – Read Only level

The default privilege level is **0**. If you want to assign Super User level access to the account, you can enter the command without **privilege 0**, as shown in the command example above.

The **password | nopassword** parameter indicates whether the user must enter a password. If you specify **password**, enter the string for the user's password. You can enter up to 255 characters for <password-string>.

NOTE: You must be logged on with Super User access (privilege level 0) to add user accounts or configure other access parameters.

To display user account information, enter the following command:

```
BigIron(config)# show users
```

Syntax: show users

Note About Changing Local User Passwords

Starting with release 03.3.00 for the FES, release 02.0.00 for the FastIron SuperX, and Enterprise release 07.8.00, the Foundry device stores not only the current password configured for a local user, but the previous two passwords configured for the user as well. The local user's password cannot be changed to one of the stored passwords.

Consequently, if you change the password for a local user, you must select a password that is different from the current password, as well as different from the previous two passwords that had been configured for that user.

For example, say local user waldo originally had a password of "whereis", and the password was subsequently changed to “whois”, then later changed to “whyis”. If you change waldo's password again, you cannot change it to "whereis", "whois", or "whyis".

The current and previous passwords are stored in the device’s running-config file in encrypted form. For example:

```
BigIron# show run
...
username waldo password 8 $1$Ro2..Ox0$udBu7pQT5XyuaXMUiUHy9. history
$1$eq...T62$IfpxIcxnDWX7CSVQKIodu. $1$QD3..2Q0$DYxgxCI64ZOSsYmSSaA28/
...
```

In the running-config file, the user’s previous two passwords are displayed in encrypted form following the **history** parameter.

USING THE WEB MANAGEMENT INTERFACE

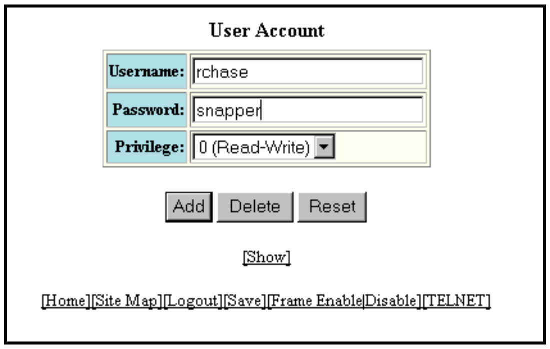
To configure a local user account using the Web management interface, use the following procedure.

NOTE: Before you can add a local user account using the Web management interface, you must enable this capability by entering the **password any** command at the global CONFIG level of the CLI.

1. Log on to the device using a valid user name and password for read-write access.
2. Select the Management link from the System configuration panel to display the Management configuration panel.
3. Select the User Account link.
    - If any user accounts are already configured on the device, the account information is listed in a table. Select the Add User Account link to display the following panel. Notice that the password display is encrypted. If you want the passwords to be displayed in clear text, you can use the CLI to disable

encryption of password displays. See “Disabling Password Encryption” on page 2-18.

- If the device does not have any user accounts configured, the following panel is displayed.



User Account

Username:	rchase
Password:	snapper
Privilege:	0 (Read-Write)

Add
Delete
Reset
[Show]
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

4. Enter the user name in the User Name field. The name cannot contain blanks.
5. Enter the password in the Password field. The password cannot contain blanks.
6. Select the management privilege level from the Privilege pulldown menu. You can select one of the following:
   - 0 (Read-Write) – equivalent to Super User level access. The user can display and configure everything.
   - 4 (Port-Config) – allows the user to configure port parameters but not global parameters.
   - 5 (Read-Only) – allows the user to display information but not to make configuration changes.
7. Click the Add button to save the change to the device’s running-config file.
8. Repeat steps 4 – 7 for each user account. You can add up to 16 accounts.
9. Select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device’s flash memory.

Configuring SSL Security for the Web Management Interface

NOTE: The BigIron MG8 and NetIron 40G (starting with release 02.1.00), the NetIron IMR 640 (starting with release 02.1.00), IronCore or JetCore devices (starting with release 07.8.00), FES (running release 03.3.00), and FastIron SuperX (running release 02.1.00) support Secure Sockets Layer (SSL) for configuring the device using the Web Management interface.

When enabled, the SSL protocol uses digital certificates and public-private key pairs to establish a secure connection to the Foundry device. Digital certificates serve to prove the identity of a connecting client, and public-private key pairs provide a means to encrypt data sent between the device and the client.

Configuring SSL for the Web management interface consists of the following tasks:

- Enabling the SSL server on the Foundry device
- Importing an RSA certificate and private key file from a client (optional)
- Generating a certificate

Enabling the SSL Server on the Foundry Device

To enable the SSL server on the Foundry device, enter the following command:

```
BigIron(config)# web-management https
```

Syntax: [no] web-management http | https

You can enable either the HTTP or HTTPs servers with this command. You can disable both the HTTP and HTTPs servers by entering the following command:

```
BigIron(config)# no web-management
```

Syntax: no web-management

Specifying a Port for SSL Communication

By default, SSL protocol exchanges occur on TCP port 443. You can optionally change the port number used for SSL communication.

For example, the following command causes the device to use TCP port 334 for SSL communication:

```
BigIron(config)# ip ssl port 334
```

Syntax: [no] ip ssl port <port-number>

The default port for SSL communication is 443.

Importing Digital Certificates and RSA Private Key Files

To allow a client to communicate with the BigIron MG8 and NetIron 40G or other Foundry device using an SSL connection, you configure a set of digital certificates and RSA public-private key pairs on the device. A digital certificate is used for identifying the connecting client to the server. It contains information about the issuing Certificate Authority, as well as a public key. You can either import digital certificates and private keys from a server, or you can allow the Foundry device to create them.

If you want to allow the Foundry device to create the digital certificates, see the next section, “Generating an SSL Certificate”. If you choose to import an RSA certificate and private key file from a client, you can use TFTP to transfer the files.

For example, to import a digital certificate using TFTP, enter a command such as the following:

```
BigIron(config)# ip ssl certificate-data-file tftp 192.168.9.210 certfile
```

Syntax: [no] ip ssl certificate-data-file tftp <ip-addr> <certificate-filename>

NOTE: If you import a digital certificate from a client, it can be no larger than 2048 bytes.

To import an RSA private key from a client using TFTP, enter a command such as the following:

```
BigIron(config)# ip ssl private-key-file tftp 192.168.9.210 keyfile
```

Syntax: [no] ip ssl private-key-file tftp <ip-addr> <key-filename>

The <ip-addr> is the IP address of a TFTP server that contains the digital certificate or private key.

Generating an SSL Certificate

After you have imported the digital certificate, generate the SSL certificate by entering the following command:

```
BigIron(config)# crypto-ssl certificate generate
```

Syntax: [no] crypto-ssl certificate generate

If you did not already import a digital certificate from a client, the device can create a default certificate. To do this, enter the following command:

```
BigIron(config)# crypto-ssl certificate generate default
```

Syntax: [no] crypto-ssl certificate generate default

Deleting the SSL Certificate

To delete the SSL certificate, enter the following command:

```
BigIron(config)# crypto-ssl certificate zeroize
```

Syntax: [no] crypto-ssl certificate zeroize

Configuring TACACS/TACACS+ Security

You can use the security protocol Terminal Access Controller Access Control System (TACACS) or TACACS+ to authenticate the following kinds of access to the Foundry device

- Telnet access
- SSH access
- Web management access
- Access to the Privileged EXEC level and CONFIG levels of the CLI

NOTE: You cannot authenticate IronView Network Manager (SNMP) access to a Foundry device using TACACS/ TACACS+.

The TACACS and TACACS+ protocols define how authentication, authorization, and accounting information is sent between a Foundry device and an authentication database on a TACACS/TACACS+ server. TACACS/ TACACS+ services are maintained in a database, typically on a UNIX workstation or PC with a TACACS/ TACACS+ server running.

How TACACS+ Differs from TACACS

TACACS is a simple UDP-based access control protocol originally developed by BBN for MILNET. TACACS+ is an enhancement to TACACS and uses TCP to ensure reliable delivery.

TACACS+ is an enhancement to the TACACS security protocol. TACACS+ improves on TACACS by separating the functions of authentication, authorization, and accounting (AAA) and by encrypting all traffic between the Foundry device and the TACACS+ server. TACACS+ allows for arbitrary length and content authentication exchanges, which allow any authentication mechanism to be utilized with the Foundry device. TACACS+ is extensible to provide for site customization and future development features. The protocol allows the Foundry device to request very precise access control and allows the TACACS+ server to respond to each component of that request.

NOTE: TACACS+ provides for authentication, authorization, and accounting, but an implementation or configuration is not required to employ all three.

TACACS/TACACS+ Authentication, Authorization, and Accounting

When you configure a Foundry device to use a TACACS/TACACS+ server for authentication, the device prompts users who are trying to access the CLI for a user name and password, then verifies the password with the TACACS/TACACS+ server.

If you are using TACACS+, Foundry recommends that you also configure ***authorization***, in which the Foundry device consults a TACACS+ server to determine which management privilege level (and which associated set of commands) an authenticated user is allowed to use. You can also optionally configure ***accounting***, which causes the Foundry device to log information on the TACACS+ server when specified events occur on the device.

NOTE: By default, a user logging into the device via Telnet or SSH would first enter the User EXEC level. The user can enter the **enable** command to get to the Privileged EXEC level.

Starting with release 07.1.00, a user that is successfully authenticated can be automatically placed at the Privileged EXEC level after login. See “Entering Privileged EXEC Mode After a Telnet or SSH Login” on page 2-31.

TACACS Authentication

NOTE: Also, multiple challenges are supported for TACACS+ login authentication.

When TACACS authentication takes place, the following events occur:

1. A user attempts to gain access to the Foundry device by doing one of the following:
   - Logging into the device using Telnet, SSH, or the Web management interface
   - Entering the Privileged EXEC level or CONFIG level of the CLI
2. The user is prompted for a username and password.
3. The user enters a username and password.
4. The Foundry device sends a request containing the username and password to the TACACS server.
5. The username and password are validated in the TACACS server's database.
6. If the password is valid, the user is authenticated.

TACACS+ Authentication

When TACACS+ authentication takes place, the following events occur:

1. A user attempts to gain access to the Foundry device by doing one of the following:
   - Logging into the device using Telnet, SSH, or the Web management interface
   - Entering the Privileged EXEC level or CONFIG level of the CLI
2. The user is prompted for a username.
3. The user enters a username.
4. The Foundry device obtains a password prompt from a TACACS+ server.
5. The user is prompted for a password.
6. The user enters a password.
7. The Foundry device sends the password to the TACACS+ server.
8. The password is validated in the TACACS+ server's database.
9. If the password is valid, the user is authenticated.

TACACS+ Authorization

Foundry devices support two kinds of TACACS+ authorization:

- Exec authorization determines a user's privilege level when they are authenticated
- Command authorization consults a TACACS+ server to get authorization for commands entered by the user

When TACACS+ exec authorization takes place, the following events occur:

1. A user logs into the Foundry device using Telnet, SSH, or the Web management interface
2. The user is authenticated.
3. The Foundry device consults the TACACS+ server to determine the privilege level of the user.
4. The TACACS+ server sends back a response containing an A-V (Attribute-Value) pair with the privilege level of the user.
5. The user is granted the specified privilege level.

When TACACS+ command authorization takes place, the following events occur:

1. A Telnet, SSH, or Web management interface user previously authenticated by a TACACS+ server enters a command on the Foundry device.
2. The Foundry device looks at its configuration to see if the command is at a privilege level that requires TACACS+ command authorization.
3. If the command belongs to a privilege level that requires authorization, the Foundry device consults the TACACS+ server to see if the user is authorized to use the command.

4. If the user is authorized to use the command, the command is executed.

TACACS+ Accounting

TACACS+ accounting works as follows:

1. One of the following events occur on the Foundry device:
   - A user logs into the management interface using Telnet or SSH
   - A user enters a command for which accounting has been configured
   - A system event occurs, such as a reboot or reloading of the configuration file
2. The Foundry device checks its configuration to see if the event is one for which TACACS+ accounting is required.
3. If the event requires TACACS+ accounting, the Foundry device sends a TACACS+ Accounting Start packet to the TACACS+ accounting server, containing information about the event.
4. The TACACS+ accounting server acknowledges the Accounting Start packet.
5. The TACACS+ accounting server records information about the event.
6. When the event is concluded, the Foundry device sends an Accounting Stop packet to the TACACS+ accounting server.
7. The TACACS+ accounting server acknowledges the Accounting Stop packet.

AAA Operations for TACACS/TACACS+

The following table lists the sequence of authentication, authorization, and accounting operations that take place when a user gains access to a Foundry device that has TACACS/TACACS+ security configured.

User Action	Applicable AAA Operations
User attempts to gain access to the Privileged EXEC and CONFIG levels of the CLI	Enable authentication: aaa authentication enable default <method-list>
	Exec authorization (TACACS+): aaa authorization exec default tacacs+
	System accounting start (TACACS+): aaa accounting system default start-stop <method-list>
User logs in using Telnet/SSH	Login authentication: aaa authentication login default <method-list>
	Exec authorization (TACACS+): aaa authorization exec default tacacs+

	Exec accounting start (TACACS+):
	aaa accounting exec default <method-list>
	System accounting start (TACACS+):
	aaa accounting system default start-stop <method-list>

User Action	Applicable AAA Operations
User logs into the Web management interface	Web authentication: aaa authentication web-server default <method-list>
	Exec authorization (TACACS+): aaa authorization exec default tacacs+

User logs out of Telnet/SSH session	Command accounting (TACACS+):
	aaa accounting commands <privilege-level> default start-stop <method-list>
	EXEC accounting stop (TACACS+):
	aaa accounting exec default start-stop <method-list>

User enters system commands (for example, **reload**, **boot system**)	Command authorization (TACACS+): aaa authorization commands <privilege-level> default <method-list>

	Command accounting (TACACS+):
	aaa accounting commands <privilege-level> default start-stop <method-list>
	System accounting stop (TACACS+):
	aaa accounting system default start-stop <method-list>

User enters the command: [no] aaa accounting system default start-stop <method-list>	Command authorization (TACACS+): aaa authorization commands <privilege-level> default <method-list>

	Command accounting (TACACS+):
	aaa accounting commands <privilege-level> default start-stop <method-list>
	System accounting start (TACACS+):
	aaa accounting system default start-stop <method-list>

User enters other commands	Command authorization (TACACS+): aaa authorization commands <privilege-level> default <method-list>
	Command accounting (TACACS+): aaa accounting commands <privilege-level> default start-stop <method-list>

AAA Security for Commands Pasted Into the Running-Config

If AAA security is enabled on the device, commands pasted into the running-config are subject to the same AAA operations as if they were entered manually.

When you paste commands into the running-config, and AAA command authorization and/or accounting is configured on the device, AAA operations are performed on the pasted commands. The AAA operations are performed before the commands are actually added to the running-config. The server performing the AAA operations should be reachable when you paste the commands into the running-config file. If the device determines that a pasted command is invalid, AAA operations are halted on the remaining commands. The remaining commands may not be executed if command authorization is configured.

TACACS/TACACS+ Configuration Considerations

- You must deploy at least one TACACS/TACACS+ server in your network.
- Foundry devices support authentication using up to eight TACACS/TACACS+ servers. The device tries to use the servers in the order you add them to the device's configuration.
- You can select only one primary authentication method for each type of access to a device (CLI through Telnet, CLI Privileged EXEC and CONFIG levels). For example, you can select TACACS+ as the primary authentication method for Telnet CLI access, but you cannot also select RADIUS authentication as a primary method for the same type of access. However, you can configure backup authentication methods for each access type.
- You can configure the Foundry device to authenticate using a TACACS or TACACS+ server, not both.

TACACS Configuration Procedure

For TACACS configurations, use the following procedure:

1. Identify TACACS servers. See "Identifying the TACACS/TACACS+ Servers" on page 2-27.
2. Set optional parameters. See "Setting Optional TACACS/TACACS+ Parameters" on page 2-28.
3. Configure authentication-method lists. See "Configuring Authentication-Method Lists for TACACS/TACACS+" on page 2-29.

TACACS+ Configuration Procedure

For TACACS+ configurations, use the following procedure:

1. Identify TACACS+ servers. See "Identifying the TACACS/TACACS+ Servers" on page 2-27.
2. Set optional parameters. See "Setting Optional TACACS/TACACS+ Parameters" on page 2-28.
3. Configure authentication-method lists. See "Configuring Authentication-Method Lists for TACACS/TACACS+" on page 2-29.
4. Optionally configure TACACS+ authorization. See "Configuring TACACS+ Authorization" on page 2-32.
5. Optionally configure TACACS+ accounting. See "Configuring TACACS+ Accounting" on page 2-34.

Identifying the TACACS/TACACS+ Servers

To use TACACS/TACACS+ servers to authenticate access to a Foundry device, you must identify the servers to the Foundry device.

For example, to identify three TACACS/TACACS+ servers, enter commands such as the following:

```
BigIron(config)# tacacs-server host 207.94.6.161
BigIron(config)# tacacs-server host 207.94.6.191
BigIron(config)# tacacs-server host 207.94.6.122
```

Syntax: tacacs-server <ip-addr>|<hostname> [auth-port <number>]

The <ip-addr>|<hostname> parameter specifies the IP address or host name of the server. You can enter up to eight **tacacs-server host** commands to specify up to eight different servers.

NOTE: To specify the server's host name instead of its IP address, you must first identify a DNS server using the **ip dns server-address** <ip-addr> command at the global CONFIG level.

If you add multiple TACACS/TACACS+ authentication servers to the Foundry device, the device tries to reach them in the order you add them. For example, if you add three servers in the following order, the software tries the servers in the same order:

1. 207.94.6.161
2. 207.94.6.191
3. 207.94.6.122

You can remove a TACACS/TACACS+ server by entering **no** followed by the **tacacs-server** command. For example, to remove 207.94.6.161, enter the following command:

```
BigIron(config)# no tacacs-server host 207.94.6.161
```

NOTE: If you erase a **tacacs-server** command (by entering "**no**" followed by the command), make sure you also erase the **aaa** commands that specify TACACS/TACACS+ as an authentication method. (See "Configuring Authentication-Method Lists for TACACS/TACACS+" on page 2-29.) Otherwise, when you exit from the CONFIG mode or from a Telnet session, the system continues to believe it is TACACS/TACACS+ enabled and you will not be able to access the system.

The **auth-port** parameter specifies the UDP (for TACACS) or TCP (for TACACS+) port number of the authentication port on the server. The default port number is 49.

Specifying Different Servers for Individual AAA Functions

In a TACACS+ configuration, you can designate a server to handle a specific AAA task. For example, you can designate one TACACS+ server to handle authorization and another TACACS+ server to handle accounting. You can set the TACACS+ key for each server.

To specify different TACACS+ servers for authentication, authorization, and accounting:

```
BigIron(config)# tacacs-server host 1.2.3.4 auth-port 49 authentication-only
key abc
BigIron(config)# tacacs-server host 1.2.3.5 auth-port 49 authorization-only key
def
BigIron(config)# tacacs-server host 1.2.3.6 auth-port 49 accounting-only key
ghi
```

Syntax: tacacs-server host <ip-addr> | <server-name> [authentication-only | authorization-only | accounting-only | default] [key <string>]

The **default** parameter causes the server to be used for all AAA functions.

After authentication takes place, the server that performed the authentication is used for authorization and/or accounting. If the authenticating server cannot perform the requested function, then the next server in the configured list of servers is tried; this process repeats until a server that can perform the requested function is found, or every server in the configured list has been tried.

Setting Optional TACACS/TACACS+ Parameters

You can set the following optional parameters in a TACACS/TACACS+ configuration:

- TACACS+ key – This parameter specifies the value that the Foundry device sends to the TACACS+ server when trying to authenticate user access.
- Retransmit interval – This parameter specifies how many times the Foundry device will resend an authentication request when the TACACS/TACACS+ server does not respond. The retransmit value can be from 1 – 5 times. The default is 3 times.
- Dead time – This parameter specifies how long the Foundry device waits for the primary authentication server to reply before deciding the server is dead and trying to authenticate using the next server. The dead-time value can be from 1 – 5 seconds. The default is 3 seconds.
- Timeout – This parameter specifies how many seconds the Foundry device waits for a response from a TACACS/TACACS+ server before either retrying the authentication request, or determining that the TACACS/TACACS+ servers are unavailable and moving on to the next authentication method in the authentication-method list. The timeout can be from 1 – 15 seconds. The default is 3 seconds.

Setting the TACACS+ Key

The **key** parameter in the **tacacs-server** command is used to encrypt TACACS+ packets before they are sent over the network. The value for the **key** parameter on the Foundry device should match the one configured on the TACACS+ server. The key can be from 1 – 32 characters in length and cannot include any space characters.

NOTE: The **tacacs-server key** command applies only to TACACS+ servers, not to TACACS servers. If you are configuring TACACS, do not configure a key on the TACACS server and do not enter a key on the Foundry device.

To specify a TACACS+ server key:

```
BigIron(config)# tacacs-server key rkwong
```

Syntax: tacacs-server key [0 | 1] <string>

When you display the configuration of the Foundry device, the TACACS+ keys are encrypted. For example:

```
BigIron(config)# tacacs-server key 1 abc
BigIron(config)# write terminal
...
tacacs-server host 1.2.3.5 auth-port 49
tacacs key 1 $!2d
```

NOTE: Encryption of the TACACS+ keys is done by default. The **0** parameter disables encryption. The **1** parameter is not required; it is provided for backwards compatibility.

Setting the Retransmission Limit

The **retransmit** parameter specifies how many times the Foundry device will resend an authentication request when the TACACS/TACACS+ server does not respond. The retransmit limit can be from 1 – 5 times. The default is 3 times.

To set the TACACS/TACACS+ retransmit limit:

```
BigIron(config)# tacacs-server retransmit 5
```

Syntax: tacacs-server retransmit <number>

Setting the Dead Time Parameter

The **dead-time** parameter specifies how long the Foundry device waits for the primary authentication server to reply before deciding the server is dead and trying to authenticate using the next server. The dead-time value can be from 1 – 5 seconds. The default is 3 seconds.

To set the TACACS/TACACS+ dead-time value:

```
BigIron(config)# tacacs-server dead-time 5
```

Syntax: tacacs-server dead-time <number>

Setting the Timeout Parameter

The **timeout** parameter specifies how many seconds the Foundry device waits for a response from the TACACS/ TACACS+ server before either retrying the authentication request, or determining that the TACACS/TACACS+ server is unavailable and moving on to the next authentication method in the authentication-method list. The timeout can be from 1 – 15 seconds. The default is 3 seconds.

```
BigIron(config)# tacacs-server timeout 5
```

Syntax: tacacs-server timeout <number>

Configuring Authentication-Method Lists for TACACS/TACACS+

You can use TACACS/TACACS+ to authenticate Telnet/SSH access and access to Privileged EXEC level and CONFIG levels of the CLI. When configuring TACACS/TACACS+ authentication, you create authentication-method lists specifically for these access methods, specifying TACACS/TACACS+ as the primary authentication method.

Within the authentication-method list, TACACS/TACACS+ is specified as the primary authentication method and up to six backup authentication methods are specified as alternates. If TACACS/TACACS+ authentication fails due to an error, the device tries the backup authentication methods in the order they appear in the list.

When you configure authentication-method lists for TACACS/TACACS+ authentication, you must create a separate authentication-method list for Telnet/SSH CLI access, and for access to the Privileged EXEC level and CONFIG levels of the CLI.

To create an authentication-method list that specifies TACACS/TACACS+ as the primary authentication method for securing Telnet/SSH access to the CLI:

```
BigIron(config)# enable telnet authentication
BigIron(config)# aaa authentication login default tacacs local
```

The commands above cause TACACS/TACACS+ to be the primary authentication method for securing Telnet/SSH access to the CLI. If TACACS/TACACS+ authentication fails due to an error with the server, authentication is performed using local user accounts instead.

To create an authentication-method list that specifies TACACS/TACACS+ as the primary authentication method for securing access to Privileged EXEC level and CONFIG levels of the CLI:

```
BigIron(config)# aaa authentication enable default tacacs local none
```

The command above causes TACACS/TACACS+ to be the primary authentication method for securing access to Privileged EXEC level and CONFIG levels of the CLI. If TACACS/TACACS+ authentication fails due to an error with the server, local authentication is used instead. If local authentication fails, no authentication is used; the device automatically permits access.

Syntax: [no] aaa authentication enable | login default <method1> [<method2>] [<method3>] [<method4>] [<method5>] [<method6>] [<method7>]

The **web-server | enable | login** parameter specifies the type of access this authentication-method list controls. You can configure one authentication-method list for each type of access.

NOTE: If you configure authentication for Web management access, authentication is performed each time a page is requested from the server. When frames are enabled on the Web management interface, the browser sends an HTTP request for each frame. The Foundry device authenticates each HTTP request from the browser. To limit authentications to one per page, disable frames on the Web management interface.

The <method1> parameter specifies the primary authentication method. The remaining optional <method> parameters specify additional methods to try if an error occurs with the primary method. A method can be one of the values listed in the Method Parameter column in the following table.

Table 2.2: Authentication Method Values

Method Parameter	Description
line	Authenticate using the password you configured for Telnet access. The Telnet password is configured using the **enable telnet password…** command. See “Setting a Telnet Password” on page 2-15.
enable	Authenticate using the password you configured for the Super User privilege level. This password is configured using the **enable super-user-password…** command. See “Setting Passwords for Management Privilege Levels” on page 2-16.

Table 2.2: Authentication Method Values (Continued)

Method Parameter	Description
local	Authenticate using a local user name and password you configured on the device. Local user names and passwords are configured using the **username…** command. See “Configuring a Local User Account” on page 2-19.
tacacs	Authenticate using the database on a TACACS server. You also must identify the server to the device using the **tacacs-server** command.
tacacs+	Authenticate using the database on a TACACS+ server. You also must identify the server to the device using the **tacacs-server** command.
radius	Authenticate using the database on a RADIUS server. You also must identify the server to the device using the **radius-server** command.
none	Do not use any authentication method. The device automatically permits access.

NOTE: For examples of how to define authentication-method lists for types of authentication other than TACACS/TACACS+, see “Configuring Authentication-Method Lists” on page 2-56.

Entering Privileged EXEC Mode After a Telnet or SSH Login

By default, a user enters User EXEC mode after a successful login through Telnet or SSH. Optionally, you can configure the device so that a user enters Privileged EXEC mode after a Telnet or SSH login. To do this, use the following command:

```
BigIron(config)# aaa authentication login privilege-mode
```

Syntax: aaa authentication login privilege-mode

The user’s privilege level is based on the privilege level granted during login.

Configuring Enable Authentication to Prompt for Password Only

If Enable authentication is configured on the device, when a user attempts to gain Super User access to the Privileged EXEC and CONFIG levels of the CLI, by default he or she is prompted for a username and password. In this release, you can configure the Foundry device to prompt only for a password. The device uses the username entered at login, if one is available. If no username was entered at login, the device prompts for both username and password.

To configure the Foundry device to prompt only for a password when a user attempts to gain Super User access to the Privileged EXEC and CONFIG levels of the CLI:

```
BigIron(config)# aaa authentication enable implicit-user
```

Syntax: [no] aaa authentication enable implicit-user

Telnet/SSH Prompts When the TACACS+ Server is Unavailable

When TACACS+ is the first method in the authentication method list, the device displays the login prompt received from the TACACS+ server. If a user attempts to login through Telnet or SSH, but none of the configured TACACS+ servers are available, the following takes place:

- If the next method in the authentication method list is "enable", the login prompt is skipped, and the user is prompted for the Enable password (that is, the password configured with the **enable super-user-password** command).
- If the next method in the authentication method list is "line", the login prompt is skipped, and the user is prompted for the Line password (that is, the password configured with the **enable telnet password** command).

Configuring TACACS+ Authorization

Foundry devices support TACACS+ authorization for controlling access to management functions in the CLI. Two kinds of TACACS+ authorization are supported:

- Exec authorization determines a user's privilege level when they are authenticated
- Command authorization consults a TACACS+ server to get authorization for commands entered by the user

Configuring Exec Authorization

When TACACS+ exec authorization is performed, the Foundry device consults a TACACS+ server to determine the privilege level of the authenticated user. To configure TACACS+ exec authorization on the Foundry device, enter the following command:

```
BigIron(config)# aaa authorization exec default tacacs+
```

Syntax: aaa authorization exec default tacacs+ | none

If you specify **none**, or omit the **aaa authorization exec** command from the device's configuration, no exec authorization is performed.

A user's privilege level is obtained from the TACACS+ server in the "foundry-privlvl" A-V pair. If the **aaa authorization exec default tacacs** command exists in the configuration, the device assigns the user the privilege level specified by this A-V pair. If the command does not exist in the configuration, then the value in the "foundry-privlvl" A-V pair is ignored, and the user is granted Super User access.

NOTE: If the **aaa authorization exec default tacacs+** command exists in the configuration, following successful authentication the device assigns the user the privilege level specified by the "foundry-privlvl" A-V pair received from the TACACS+ server. If the **aaa authorization exec default tacacs+** command does not exist in the configuration, then the value in the "foundry-privlvl" A-V pair is ignored, and the user is granted Super User access.

Also note that in order for the **aaa authorization exec default tacacs+** command to work, either the **aaa authentication enable default tacacs+** command, or the **aaa authentication login privilege-mode** command must also exist in the configuration.

Configuring an Attribute-Value Pair on the TACACS+ Server

During TACACS+ exec authorization, the Foundry device expects the TACACS+ server to send a response containing an A-V (Attribute-Value) pair that specifies the privilege level of the user. When the Foundry device receives the response, it extracts an A-V pair configured for the Exec service and uses it to determine the user's privilege level.

To set a user's privilege level, you can configure the "foundry-privlvl" A-V pair for the Exec service on the TACACS+ server. For example:

```
user=bob {
   default service = permit
   member admin
   # Global password
   global = cleartext "cat"
   service = exec {
     foundry-privlvl = 0
        }
}
```

In this example, the A-V pair `foundry-privlvl = 0` grants the user full read-write access. The value in the foundry-privlvl A-V pair is an integer that indicates the privilege level of the user. Possible values are 0 for super-user level, 4 for port-config level, or 5 for read-only level. If a value other than 0, 4, or 5 is specified in the foundry-privlvl A-V pair, the default privilege level of 5 (read-only) is used. The foundry-privlvl A-V pair can also be embedded in the group configuration for the user. See your TACACS+ documentation for the configuration syntax relevant to your server.

If the foundry-privlvl A-V pair is not present, the Foundry device extracts the last A-V pair configured for the Exec service that has a numeric value. The Foundry device uses this A-V pair to determine the user's privilege level. For example:

```
user=bob {
   default service = permit
   member admin
   # Global password
   global = cleartext "cat"
   service = exec {
     privlvl = 15
        }
}
```

The attribute name in the A-V pair is not significant; the Foundry device uses the last one that has a numeric value. However, the Foundry device interprets the value for a non-"foundry-privlvl" A-V pair differently than it does for a "foundry-privlvl" A-V pair. The following table lists how the Foundry device associates a value from a non-"foundry-privlvl" A-V pair with a Foundry privilege level.

Table 2.3: Foundry Equivalents for non-"foundry-privlvl" A-V Pair Values

Value for non-"foundry-privlvl" A-V Pair	Foundry Privilege Level
15	0 (super-user)
From 14 – 1	4 (port-config)
Any other number or 0	5 (read-only)

In the example above, the A-V pair configured for the Exec service is `privlvl = 15`. The Foundry device uses the value in this A-V pair to set the user's privilege level to 0 (super-user), granting the user full read-write access.

In a configuration that has both a "foundry-privlvl" A-V pair and a non-"foundry-privlvl" A-V pair for the Exec service, the non-"foundry-privlvl" A-V pair is ignored. For example:

```
user=bob {
   default service = permit
   member admin
   # Global password
   global = cleartext "cat"
   service = exec {
     foundry-privlvl = 4
     privlvl = 15
        }
}
```

In this example, the user would be granted a privilege level of 4 (port-config level). The `privlvl = 15` A-V pair is ignored by the Foundry device.

If the TACACS+ server has no A-V pair configured for the Exec service, the default privilege level of 5 (read-only) is used.

Configuring Command Authorization

When TACACS+ command authorization is enabled, the Foundry device consults a TACACS+ server to get authorization for commands entered by the user.

You enable TACACS+ command authorization by specifying a privilege level whose commands require authorization. For example, to configure the Foundry device to perform authorization for the commands available at the Super User privilege level (that is, all commands on the device), enter the following command:

```
BigIron(config)# aaa authorization commands 0 default tacacs+
```

Syntax: aaa authorization commands <privilege-level> default tacacs+ | radius | none

The <privilege-level> parameter can be one of the following:

- **0** – Authorization is performed for commands available at the Super User level (all commands)
- **4** – Authorization is performed for commands available at the Port Configuration level (port-config and read-only commands)
- **5** – Authorization is performed for commands available at the Read Only level (read-only commands)

NOTE: TACACS+ command authorization can be performed only for commands entered from Telnet or SSH sessions, or from the console. No authorization is performed for commands entered at the Web management interface or IronView Network Manager.

TACACS+ command authorization is not performed for the following commands:

- At all levels: **exit**, **logout**, **end**, and **quit**.
- At the Privileged EXEC level: **enable** or **enable** <text>, where <text> is the password configured for the Super User privilege level.

If configured, command accounting is performed for these commands.

AAA Support for Console Commands

In releases prior to 07.8.00 and FES 03.4.01, authentication, authorization, and accounting (AAA) support for CLI commands entered at the console was limited to command authorization and command accounting. Release 03.4.01 enhances AAA support for commands entered at the console.

Starting with this release, AAA support for commands entered at the console can include the following:

- Login prompt that uses AAA authentication, using authentication-method Lists
- Exec Authorization
- Exec Accounting
- System Accounting

To enable AAA support for commands entered at the console, enter the following command:

```
FES Switch(config)# enable aaa console
```

Syntax: [no] enable aaa console

NOTE: In previous releases, the **enable aaa console** command only enabled command authorization and command accounting for CLI commands entered at the console.

Configuring TACACS+ Accounting

Foundry devices support TACACS+ accounting for recording information about user activity and system events. When you configure TACACS+ accounting on a Foundry device, information is sent to a TACACS+ accounting server when specified events occur, such as when a user logs into the device or the system is rebooted.

Configuring TACACS+ Accounting for Telnet/SSH (Shell) Access

To send an Accounting Start packet to the TACACS+ accounting server when an authenticated user establishes a Telnet or SSH session on the Foundry device, and an Accounting Stop packet when the user logs out:

```
BigIron(config)# aaa accounting exec default start-stop tacacs+
```

Syntax: aaa accounting exec default start-stop radius | tacacs+ | none

Configuring TACACS+ Accounting for CLI Commands

You can configure TACACS+ accounting for CLI commands by specifying a privilege level whose commands require accounting. For example, to configure the Foundry device to perform TACACS+ accounting for the commands available at the Super User privilege level (that is; all commands on the device), enter the following command:

```
BigIron(config)# aaa accounting commands 0 default start-stop tacacs+
```

An Accounting Start packet is sent to the TACACS+ accounting server when a user enters a command, and an Accounting Stop packet is sent when the service provided by the command is completed.

NOTE: If authorization is enabled, and the command requires authorization, then authorization is performed before accounting takes place. If authorization fails for the command, no accounting takes place.

Syntax: aaa accounting commands <privilege-level> default start-stop radius | tacacs+ | none

The <privilege-level> parameter can be one of the following:

- **0** – Records commands available at the Super User level (all commands)
- **4** – Records commands available at the Port Configuration level (port-config and read-only commands)
- **5** – Records commands available at the Read Only level (read-only commands)

Configuring TACACS+ Accounting for System Events

You can configure TACACS+ accounting to record when system events occur on the Foundry device. System events include rebooting and when changes to the active configuration are made.

The following command causes an Accounting Start packet to be sent to the TACACS+ accounting server when a system event occurs, and a Accounting Stop packet to be sent when the system event is completed:

```
BigIron(config)# aaa accounting system default start-stop tacacs+
```

Syntax: aaa accounting system default start-stop radius | tacacs+ | none

Configuring an Interface as the Source for All TACACS/TACACS+ Packets

You can designate the lowest-numbered IP address configured an Ethernet port, POS port, loopback interface, or virtual interface as the source IP address for all TACACS/TACACS+ packets from the Layer 3 Switch. Identifying a single source IP address for TACACS/TACACS+ packets provides the following benefits:

- If your TACACS/TACACS+ server is configured to accept packets only from specific links or IP addresses, you can use this feature to simplify configuration of the TACACS/TACACS+ server by configuring the Foundry device to always send the TACACS/TACACS+ packets from the same link or source address.
- If you specify a loopback interface as the single source for TACACS/TACACS+ packets, TACACS/TACACS+ servers can receive the packets regardless of the states of individual links. Thus, if a link to the TACACS/TACACS+ server becomes unavailable but the client or server can be reached through another link, the client or server still receives the packets, and the packets still have the source IP address of the loopback interface.

The software contains separate CLI commands for specifying the source interface for Telnet, TACACS/TACACS+, and RADIUS packets. You can configure a source interface for one or more of these types of packets.

To specify an Ethernet or POS port or a loopback or virtual interface as the source for all TACACS/TACACS+ packets from the device, use the following CLI method. The software uses the lowest-numbered IP address configured on the port or interface as the source IP address for TACACS/TACACS+ packets originated by the device.

To specify the lowest-numbered IP address configured on a virtual interface as the device's source for all TACACS/TACACS+ packets, enter commands such as the following:

```
BigIron(config)# int ve 1
BigIron(config-vif-1)# ip address 10.0.0.3/24
BigIron(config-vif-1)# exit
BigIron(config)# ip tacacs source-interface ve 1
```

The commands in this example configure virtual interface 1, assign IP address 10.0.0.3/24 to the interface, then designate the interface as the source for all TACACS/TACACS+ packets from the Layer 3 Switch.

Syntax: ip tacacs source-interface ethernet <portnum> | pos <portnum> | loopback <num> | ve <num>

The <num> parameter is a loopback interface or virtual interface number. If you specify an Ethernet or POS port, the <portnum> is the port's number (including the slot number, if you are configuring a device).

Displaying TACACS/TACACS+ Statistics and Configuration Information

The **show aaa** command displays information about all TACACS+ and RADIUS servers identified on the device. For example:

```
BigIron# show aaa
Tacacs+ key: foundry
Tacacs+ retries: 1
Tacacs+ timeout: 15 seconds
Tacacs+ dead-time: 3 minutes
Tacacs+ Server: 207.95.6.90 Port:49:
                opens=6 closes=3 timeouts=3 errors=0
                packets in=4 packets out=4
no connection

Radius key: networks
Radius retries: 3
Radius timeout: 3 seconds
Radius dead-time: 3 minutes
Radius Server:  207.95.6.90 Auth Port=1645 Acct Port=1646:
                opens=2 closes=1 timeouts=1 errors=0
                packets in=1 packets out=4
no connection
```

The following table describes the TACACS/TACACS+ information displayed by the **show aaa** command.

Table 2.4: Output of the show aaa command for TACACS/TACACS+

Field	Description
Tacacs+ key	The setting configured with the **tacacs-server key** command. At the Super User privilege level, the actual text of the key is displayed. At the other privilege levels, a string of periods (....) is displayed instead of the text.
Tacacs+ retries	The setting configured with the **tacacs-server retransmit** command.
Tacacs+ timeout	The setting configured with the **tacacs-server timeout** command.
Tacacs+ dead-time	The setting configured with the **tacacs-server dead-time** command.
Tacacs+ Server	For each TACACS/TACACS+ server, the IP address, port, and the following statistics are displayed: opens — Number of times the port was opened for communication with the server closes — Number of times the port was closed normally timeouts — Number of times port was closed due to a timeout errors — Number of times an error occurred while opening the port packets in — Number of packets received from the server packets out — Number of packets sent to the server

Table 2.4: Output of the show aaa command for TACACS/TACACS+

Field	Description
connection	The current connection status. This can be “no connection” or “connection active”.

The **show web** command displays the privilege level of Web management interface users. For example:

```
BigIron(config)#show web
User                              Privilege       IP address
set                                      0        192.168.1.234
```

Syntax: show web

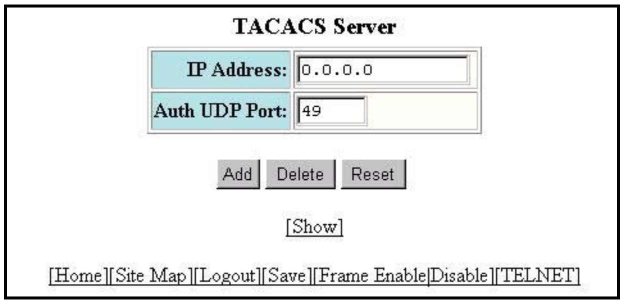
USING THE WEB MANAGEMENT INTERFACE

To configure TACACS/TACACS+ using the Web management interface:

1. Log on to the device using a valid user name and password for read-write access. The System configuration panel is displayed.
2. If you configuring TACACS/TACACS+ authentication for Telnet access to the CLI, go to step 3. Otherwise, go to step 7.
3. Select the Management link to display the Management configuration panel.
4. Select Enable next to Telnet Authentication. You must enable Telnet authentication if you want to use TACACS/TACACS+ or RADIUS to authenticate Telnet access to the device.
5. Click Apply to apply the change.
6. Select the Home link to return to the System configuration panel.
7. Select the TACACS link from the System configuration panel to display the TACACS panel.
8. If needed, change the Authentication port and Accounting port. (The default values work in most networks.)
9. Enter the key if applicable.

NOTE: The **key** parameter applies only to TACACS+ servers, not to TACACS servers. If you are configuring for TACACS authentication, do not configure a key on the TACACS server and do not enter a key on the Foundry device.

10. Click Apply if you changed any TACACS/TACACS+ parameters.
11. Select the TACACS Server link.
    - If any TACACS/TACACS+ servers are already configured on the device, the servers are listed in a table. Select the Add TACACS Server link to display the TACACS configuration panel.
    - If the device does not have any TACACS servers configured, the following panel is displayed.



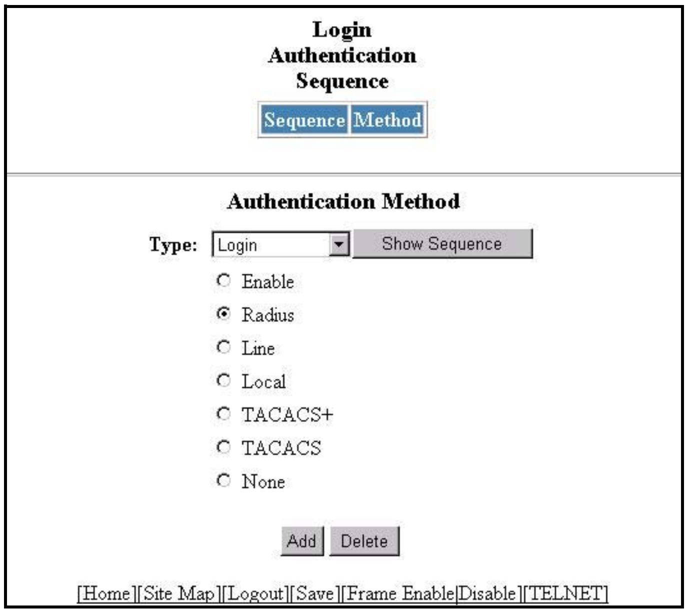
TACACS Server

IP Address: 0.0.0.0

Auth UDP Port: 49

Add
Delete
Reset
[Show]
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

12. Enter the server's IP address in the IP Address field.
13. If needed, change the Authentication port and Accounting port. (The default values work in most networks.)
14. Click Home to return to the System configuration panel, then select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.
15. Select the Management link to display the Management configuration panel.
16. Select the Authentication Methods link to display the Login Authentication Sequence panel, as shown in the following example.


Login Authentication Sequence
Sequence
Method
Show Sequence
Add
Delete
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

Authentication Method

Type: Login

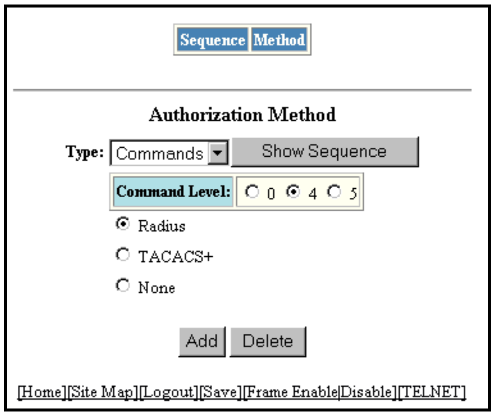
- ( ) Enable
- (•) Radius
- ( ) Line
- ( ) Local
- ( ) TACACS+
- ( ) TACACS
- ( ) None

17. Select the type of access for which you are defining the authentication method list from the Type field's pulldown menu. Each type of access must have a separate authentication-method list. For example, to define the authentication-method list for logging into the CLI, select Login.
18. Select the primary authentication method by clicking on the radio button next to the method. For example, to use a TACACS+ server as the primary means of authentication for logging on to the CLI, select TACACS+.
19. Click the Add button to save the change to the device's running-config file.

    The access type and authentication method you selected are displayed in the table at the top of the dialog. Each time you add an authentication method for a given access type, the software assigns a sequence number to the entry. When the user tries to log in using the access type you selected, the software tries the authentication sources in ascending sequence order until the access request is either approved or denied. Each time you add an entry for a given access type, the software increments the sequence number. Thus, if you want to use multiple authentication methods, make sure you enter the primary authentication method first, the secondary authentication method second, and so on.

    If you need to delete an entry, select the access type and authentication method for the entry, then click Delete.
20. Click Home to return to the System configuration panel, then select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

21. To configure TACACS+ authorization, select the Management link to display the Management configuration panel and select the Authorization Methods link to display the Authorization Method panel, as shown in the following example.



22. To configure TACACS+ exec authorization, select Exec from the Type field's pulldown menu.
23. To configure TACACS+ command authorization, select Commands from the Type field's pulldown menu and select a privilege level by clicking on one of the following radio buttons:
    - **0** – Authorization is performed for commands available at the Super User level (all commands)
    - **4** – Authorization is performed for commands available at the Port Configuration level (port-config and read-only commands)
    - **5** – Authorization is performed for commands available at the Read Only level (read-only commands)

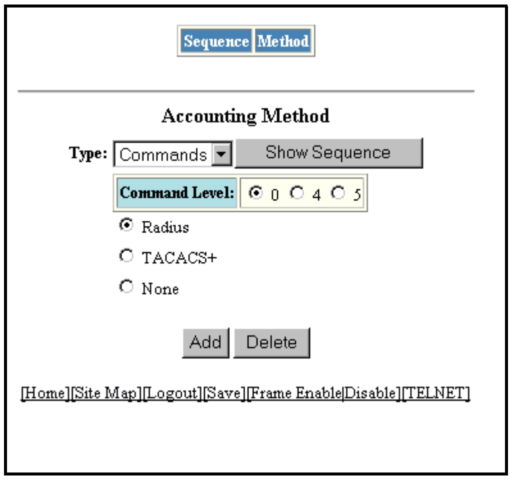
NOTE: TACACS+ command authorization can be performed only for commands entered from Telnet or SSH sessions, or from the console. No authorization is performed for commands entered at the Web management interface or IronView Network Manager.

24. Click on the radio button next to TACACS+.
25. Click the Add button to save the change to the device's running-config file.

    The authorization method you selected are displayed in the table at the top of the dialog. Each time you add an authorization method for a given access type, the software assigns a sequence number to the entry. When authorization is performed, the software tries the authorization sources in ascending sequence order until the request is either approved or denied. Each time you add an entry for a given access type, the software increments the sequence number. Thus, if you want to use multiple authentication methods, make sure you enter the primary authentication method first, the secondary authentication method second, and so on.

    If you need to delete an entry, select the access type and authorization method for the entry, then click Delete.

26. To configure TACACS+ accounting, select the Management link to display the Management configuration panel and select the Accounting Methods link to display the Accounting Method panel, as shown in the following example.



Accounting Method

Type: Commands

Command Level: (● 0) ○ 4 ○ 5

(● Radius)

○ TACACS+

○ None

Sequence
Method
Show Sequence
Add
Delete
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

27. To send an Accounting Start packet to the TACACS+ accounting server when an authenticated user establishes a Telnet or SSH session on the Foundry device, and an Accounting Stop packet when the user logs out, select Exec from the Type field's pulldown menu.
28. To configure TACACS+ accounting for CLI commands, select Commands from the Type field's pulldown menu and select a privilege level by clicking on one of the following radio buttons:
    - **0** – Records commands available at the Super User level (all commands)
    - **4** – Records commands available at the Port Configuration level (port-config and read-only commands)
    - **5** – Records commands available at the Read Only level (read-only commands)
29. To configure TACACS+ accounting to record when system events occur on the Foundry device, select System from the Type field's pulldown menu.
30. Click on the radio button next to TACACS+.
31. Click the Add button to save the change to the device's running-config file.

    The accounting method you selected are displayed in the table at the top of the dialog. Each time you add an accounting method for a given access type, the software assigns a sequence number to the entry. When accounting is performed, the software tries the accounting sources in ascending sequence order until the request is either approved or denied. Each time you add an entry for a given access type, the software increments the sequence number. Thus, if you want to use multiple accounting methods, make sure you enter the primary accounting method first, the secondary accounting method second, and so on.

    If you need to delete an entry, select the access type and accounting method for the entry, then click Delete.
32. Select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

Configuring RADIUS Security

You can use a Remote Authentication Dial In User Service (RADIUS) server to secure the following types of access to the Foundry Layer 2 Switch or Layer 3 Switch:

- Telnet access
- SSH access
- Web management access

- Access to the Privileged EXEC level and CONFIG levels of the CLI

NOTE: Foundry devices do not support RADIUS security for SNMP (IronView Network Manager) access.

RADIUS Authentication, Authorization, and Accounting

When RADIUS ***authentication*** is implemented, the Foundry device consults a RADIUS server to verify user names and passwords. You can optionally configure RADIUS ***authorization***, in which the Foundry device consults a list of commands supplied by the RADIUS server to determine whether a user can execute a command he or she has entered, as well as ***accounting***, which causes the Foundry device to log information on a RADIUS accounting server when specified events occur on the device.

NOTE: By default, a user logging into the device via Telnet or SSH first enters the User EXEC level. The user can then enter the **enable** command to get to the Privileged EXEC level.

Starting with release 07.1.00, a user that is successfully authenticated can be automatically placed at the Privileged EXEC level after login. See "Entering Privileged EXEC Mode After a Telnet or SSH Login" on page 2-48.

RADIUS Authentication

When RADIUS authentication takes place, the following events occur:

1. A user attempts to gain access to the Foundry device by doing one of the following:
   - Logging into the device using Telnet, SSH, or the Web management interface
   - Entering the Privileged EXEC level or CONFIG level of the CLI
2. The user is prompted for a username and password.
3. The user enters a username and password.
4. The Foundry device sends a RADIUS Access-Request packet containing the username and password to the RADIUS server.
5. The RADIUS server validates the Foundry device using a shared secret (the RADIUS key).
6. The RADIUS server looks up the username in its database.
7. If the username is found in the database, the RADIUS server validates the password.
8. If the password is valid, the RADIUS server sends an Access-Accept packet to the Foundry device, authenticating the user. Within the Access-Accept packet are three Foundry vendor-specific attributes that indicate:
   - The privilege level of the user
   - A list of commands
   - Whether the user is allowed or denied usage of the commands in the list

   The last two attributes are used with RADIUS authorization, if configured.
9. The user is authenticated, and the information supplied in the Access-Accept packet for the user is stored on the Foundry device. The user is granted the specified privilege level. If you configure RADIUS authorization, the user is allowed or denied usage of the commands in the list.

RADIUS Authorization

When RADIUS authorization takes place, the following events occur:

1. A user previously authenticated by a RADIUS server enters a command on the Foundry device.
2. The Foundry device looks at its configuration to see if the command is at a privilege level that requires RADIUS command authorization.

3. If the command belongs to a privilege level that requires authorization, the Foundry device looks at the list of commands delivered to it in the RADIUS Access-Accept packet when the user was authenticated. (Along with the command list, an attribute was sent that specifies whether the user is permitted or denied usage of the commands in the list.)

   NOTE: After RADIUS authentication takes place, the command list resides on the Foundry device. The RADIUS server is not consulted again once the user has been authenticated. This means that any changes made to the user's command list on the RADIUS server are not reflected until the next time the user is authenticated by the RADIUS server, and the new command list is sent to the Foundry device.

4. If the command list indicates that the user is authorized to use the command, the command is executed.

RADIUS Accounting

RADIUS accounting works as follows:

1. One of the following events occur on the Foundry device:
   - A user logs into the management interface using Telnet or SSH
   - A user enters a command for which accounting has been configured
   - A system event occurs, such as a reboot or reloading of the configuration file
2. The Foundry device checks its configuration to see if the event is one for which RADIUS accounting is required.
3. If the event requires RADIUS accounting, the Foundry device sends a RADIUS Accounting Start packet to the RADIUS accounting server, containing information about the event.
4. The RADIUS accounting server acknowledges the Accounting Start packet.
5. The RADIUS accounting server records information about the event.
6. When the event is concluded, the Foundry device sends an Accounting Stop packet to the RADIUS accounting server.
7. The RADIUS accounting server acknowledges the Accounting Stop packet.

AAA Operations for RADIUS

The following table lists the sequence of authentication, authorization, and accounting operations that take place when a user gains access to a Foundry device that has RADIUS security configured.

User Action	Applicable AAA Operations
User attempts to gain access to the Privileged EXEC and CONFIG levels of the CLI	Enable authentication: aaa authentication enable default <method-list>
	System accounting start: aaa accounting system default start-stop <method-list>
User logs in using Telnet/SSH	Login authentication: aaa authentication login default <method-list>
	EXEC accounting Start: aaa accounting exec default start-stop <method-list> System accounting Start: aaa accounting system default start-stop <method-list>

User Action	Applicable AAA Operations
User logs into the Web management interface	Web authentication: aaa authentication web-server default <method-list>
User logs out of Telnet/SSH session	Command authorization for **logout** command: aaa authorization commands <privilege-level> default <method-list>
	Command accounting: aaa accounting commands <privilege-level> default start-stop <method-list> EXEC accounting stop: aaa accounting exec default start-stop <method-list>
User enters system commands (for example, **reload**, **boot system**)	Command authorization: aaa authorization commands <privilege-level> default <method-list>
	Command accounting: aaa accounting commands <privilege-level> default start-stop <method-list> System accounting stop: aaa accounting system default start-stop <method-list>
User enters the command: [no] aaa accounting system default start-stop <method-list>	Command authorization: aaa authorization commands <privilege-level> default <method-list>
	Command accounting: aaa accounting commands <privilege-level> default start-stop <method-list> System accounting start: aaa accounting system default start-stop <method-list>
User enters other commands	Command authorization: aaa authorization commands <privilege-level> default <method-list>
	Command accounting: aaa accounting commands <privilege-level> default start-stop <method-list>

AAA Security for Commands Pasted Into the Running-Config

If AAA security is enabled on the device, commands pasted into the running-config are subject to the same AAA operations as if they were entered manually.

When you paste commands into the running-config, and AAA command authorization and/or accounting is configured on the device, AAA operations are performed on the pasted commands. The AAA operations are performed before the commands are actually added to the running-config. The server performing the AAA operations should be reachable when you paste the commands into the running-config file. If the device determines that a pasted command is invalid, AAA operations are halted on the remaining commands. The remaining commands may not be executed if command authorization is configured.

NOTE: Since RADIUS command authorization relies on a list of commands received from the RADIUS server when authentication is performed, it is important that you use RADIUS authentication when you also use RADIUS command authorization.

RADIUS Configuration Considerations

- You must deploy at least one RADIUS server in your network.
- Foundry devices support authentication using up to eight RADIUS servers. The device tries to use the servers in the order you add them to the device's configuration. If one RADIUS server is not responding, the Foundry device tries the next one in the list.
- You can select only one primary authentication method for each type of access to a device (CLI through Telnet, CLI Privileged EXEC and CONFIG levels). For example, you can select RADIUS as the primary authentication method for Telnet CLI access, but you cannot also select TACACS+ authentication as the primary method for the same type of access. However, you can configure backup authentication methods for each access type.

RADIUS Configuration Procedure

Use the following procedure to configure a Foundry device for RADIUS:

1. Configure Foundry vendor-specific attributes on the RADIUS server. See "Configuring Foundry-Specific Attributes on the RADIUS Server" on page 2-44.
2. Identify the RADIUS server to the Foundry device. See "Identifying the RADIUS Server to the Foundry Device" on page 2-46.
3. Set RADIUS parameters. See "Setting RADIUS Parameters" on page 2-46.
4. Configure authentication-method lists. See "Configuring Authentication-Method Lists for RADIUS" on page 2-47.
5. Optionally configure RADIUS authorization. See "Configuring RADIUS Authorization" on page 2-49.
6. Optionally configure RADIUS accounting. "Configuring RADIUS Accounting" on page 2-50.

Configuring Foundry-Specific Attributes on the RADIUS Server

NOTE: For all Foundry devices, RADIUS Challenge is supported for 802.1x authentication but not for login authentication.

During the RADIUS authentication process, if a user supplies a valid username and password, the RADIUS server sends an Access-Accept packet to the Foundry device, authenticating the user. Within the Access-Accept packet are three Foundry vendor-specific attributes that indicate:

- The privilege level of the user
- A list of commands
- Whether the user is allowed or denied usage of the commands in the list

You must add these three Foundry vendor-specific attributes to your RADIUS server's configuration, and configure the attributes in the individual or group profiles of the users that will access the Foundry device.

Foundry's Vendor-ID is 1991, with Vendor-Type 1. The following table describes the Foundry vendor-specific attributes.

Table 2.5: Foundry vendor-specific attributes for RADIUS

Attribute Name	Attribute ID	Data Type	Description
foundry-privilege-level	1	integer	Specifies the privilege level for the user. This attribute can be set to one of the following: **0** Super User level – Allows complete read-and-write access to the system. This is generally for system administrators and is the only management privilege level that allows you to configure passwords. **4** Port Configuration level – Allows read-and-write access for specific ports but not for global (system-wide) parameters. **5** Read Only level – Allows access to the Privileged EXEC mode and CONFIG mode of the CLI but only with read access.
foundry-command-string	2	string	Specifies a list of CLI commands that are permitted or denied to the user when RADIUS authorization is configured. The commands are delimited by semi-colons (;). You can specify an asterisk (*) as a wildcard at the end of a command string. For example, the following command list specifies all **show** and **debug ip** commands, as well as the **write terminal** command: show *; debug ip *; write term*
foundry-command-exception-flag	3	integer	Specifies whether the commands indicated by the foundry-command-string attribute are permitted or denied to the user. This attribute can be set to one of the following: **0** Permit execution of the commands indicated by foundry-command-string, deny all other commands. **1** Deny execution of the commands indicated by foundry-command-string, permit all other commands.

Identifying the RADIUS Server to the Foundry Device

To use a RADIUS server to authenticate access to a Foundry device, you must identify the server to the Foundry device. For example:

```
BigIron(config)# radius-server host 209.157.22.99
```

Syntax: radius-server host <ip-addr> | <server-name> [auth-port <number> acct-port <number>]

The **host** <ip-addr> **|** <server-name> parameter is either an IP address or an ASCII text string.

The <auth-port> parameter is the Authentication port number; it is an optional parameter. The default is 1645.

The <acct-port> parameter is the Accounting port number; it is an optional parameter. The default is 1646.

Specifying Different Servers for Individual AAA Functions

In a RADIUS configuration, you can designate a server to handle a specific AAA task. For example, you can designate one RADIUS server to handle authorization and another RADIUS server to handle accounting. You can specify individual servers for authentication and accounting, but not for authorization. You can set the RADIUS key for each server.

To specify different RADIUS servers for authentication, authorization, and accounting:

```
BigIron(config)# radius-server host 1.2.3.4 authentication-only key abc
BigIron(config)# radius-server host 1.2.3.5 authorization-only key def
BigIron(config)# radius-server host 1.2.3.6 accounting-only key ghi
```

Syntax: radius-server host <ip-addr> | <server-name> [authentication-only | accounting-only | default] [key 0 | 1 <string>]

The **default** parameter causes the server to be used for all AAA functions.

After authentication takes place, the server that performed the authentication is used for authorization and/or accounting. If the authenticating server cannot perform the requested function, then the next server in the configured list of servers is tried; this process repeats until a server that can perform the requested function is found, or every server in the configured list has been tried.

Setting RADIUS Parameters

You can set the following parameters in a RADIUS configuration:

- RADIUS key – This parameter specifies the value that the Foundry device sends to the RADIUS server when trying to authenticate user access.
- Retransmit interval – This parameter specifies how many times the Foundry device will resend an authentication request when the RADIUS server does not respond. The retransmit value can be from 1 – 5 times. The default is 3 times.
- Timeout – This parameter specifies how many seconds the Foundry device waits for a response from a RADIUS server before either retrying the authentication request, or determining that the RADIUS servers are unavailable and moving on to the next authentication method in the authentication-method list. The timeout can be from 1 – 15 seconds. The default is 3 seconds.

Setting the RADIUS Key

The **key** parameter in the **radius-server** command is used to encrypt RADIUS packets before they are sent over the network. The value for the **key** parameter on the Foundry device should match the one configured on the RADIUS server. The key can be from 1 – 32 characters in length and cannot include any space characters.

To specify a RADIUS server key:

```
BigIron(config)# radius-server key mirabeau
```

Syntax: radius-server key [0 | 1] <string>

When you display the configuration of the Foundry device, the RADIUS key is encrypted. For example:

```
BigIron(config)# radius-server key 1 abc
BigIron(config)# write terminal
...
radius-server host 1.2.3.5
radius key 1 $!2d
```

NOTE: Encryption of the RADIUS keys is done by default. The **0** parameter disables encryption. The **1** parameter is not required; it is provided for backwards compatibility.

Setting the Retransmission Limit

The **retransmit** parameter specifies the maximum number of retransmission attempts. When an authentication request times out, the Foundry software will retransmit the request up to the maximum number of retransmissions configured. The default retransmit value is 3 retries. The range of retransmit values is from 1 – 5.

To set the RADIUS retransmit limit:

```
BigIron(config)# radius-server retransmit 5
```

Syntax: radius-server retransmit <number>

Setting the Timeout Parameter

The **timeout** parameter specifies how many seconds the Foundry device waits for a response from the RADIUS server before either retrying the authentication request, or determining that the RADIUS server is unavailable and moving on to the next authentication method in the authentication-method list. The timeout can be from 1 – 15 seconds. The default is 3 seconds.

```
BigIron(config)# radius-server timeout 5
```

Syntax: radius-server timeout <number>

Configuring Authentication-Method Lists for RADIUS

You can use RADIUS to authenticate Telnet/SSH access and access to Privileged EXEC level and CONFIG levels of the CLI. When configuring RADIUS authentication, you create authentication-method lists specifically for these access methods, specifying RADIUS as the primary authentication method.

Within the authentication-method list, RADIUS is specified as the primary authentication method and up to six backup authentication methods are specified as alternates. If RADIUS authentication fails due to an error, the device tries the backup authentication methods in the order they appear in the list.

When you configure authentication-method lists for RADIUS, you must create a separate authentication-method list for Telnet or SSH CLI access and for CLI access to the Privileged EXEC level and CONFIG levels of the CLI.

To create an authentication-method list that specifies RADIUS as the primary authentication method for securing Telnet access to the CLI:

```
BigIron(config)# enable telnet authentication
BigIron(config)# aaa authentication login default radius local
```

The commands above cause RADIUS to be the primary authentication method for securing Telnet access to the CLI. If RADIUS authentication fails due to an error with the server, local authentication is used instead.

To create an authentication-method list that specifies RADIUS as the primary authentication method for securing access to Privileged EXEC level and CONFIG levels of the CLI:

```
BigIron(config)# aaa authentication enable default radius local none
```

The command above causes RADIUS to be the primary authentication method for securing access to Privileged EXEC level and CONFIG levels of the CLI. If RADIUS authentication fails due to an error with the server, local authentication is used instead. If local authentication fails, no authentication is used; the device automatically permits access.

Syntax: [no] aaa authentication enable | login default <method1> [<method2>] [<method3>] [<method4>] [<method5>] [<method6>] [<method7>]

The **web-server | enable | login** parameter specifies the type of access this authentication-method list controls. You can configure one authentication-method list for each type of access.

NOTE: If you configure authentication for Web management access, authentication is performed each time a page is requested from the server. When frames are enabled on the Web management interface, the browser sends an HTTP request for each frame. The Foundry device authenticates each HTTP request from the browser. To limit authentications to one per page, disable frames on the Web management interface.

The <method1> parameter specifies the primary authentication method. The remaining optional <method> parameters specify additional methods to try if an error occurs with the primary method. A method can be one of the values listed in the Method Parameter column in the following table.

Table 2.6: Authentication Method Values

Method Parameter	Description
line	Authenticate using the password you configured for Telnet access. The Telnet password is configured using the **enable telnet password…** command. See “Setting a Telnet Password” on page 2-15.
enable	Authenticate using the password you configured for the Super User privilege level. This password is configured using the **enable super-user-password…** command. See “Setting Passwords for Management Privilege Levels” on page 2-16.
local	Authenticate using a local user name and password you configured on the device. Local user names and passwords are configured using the **username…** command. See “Configuring a Local User Account” on page 2-19.
tacacs	Authenticate using the database on a TACACS server. You also must identify the server to the device using the **tacacs-server** command.
tacacs+	Authenticate using the database on a TACACS+ server. You also must identify the server to the device using the **tacacs-server** command.
radius	Authenticate using the database on a RADIUS server. You also must identify the server to the device using the **radius-server** command.
none	Do not use any authentication method. The device automatically permits access.

NOTE: For examples of how to define authentication-method lists for types of authentication other than RADIUS, see “Configuring Authentication-Method Lists” on page 2-56.

Entering Privileged EXEC Mode After a Telnet or SSH Login

By default, a user enters User EXEC mode after a successful login through Telnet or SSH. Optionally, you can configure the device so that a user enters Privileged EXEC mode after a Telnet or SSH login. To do this, use the following command:

```
BigIron(config)# aaa authentication login privilege-mode
```

Syntax: aaa authentication login privilege-mode

The user’s privilege level is based on the privilege level granted during login.

Configuring Enable Authentication to Prompt for Password Only

If Enable authentication is configured on the device, when a user attempts to gain Super User access to the Privileged EXEC and CONFIG levels of the CLI, by default he or she is prompted for a username and password. In this release, you can configure the Foundry device to prompt only for a password. The device uses the username entered at login, if one is available. If no username was entered at login, the device prompts for both username and password.

To configure the Foundry device to prompt only for a password when a user attempts to gain Super User access to the Privileged EXEC and CONFIG levels of the CLI:

```
BigIron(config)# aaa authentication enable implicit-user
```

Syntax: [no] aaa authentication enable implicit-user

Configuring RADIUS Authorization

Foundry devices support RADIUS authorization for controlling access to management functions in the CLI. Two kinds of RADIUS authorization are supported:

- Exec authorization determines a user's privilege level when they are authenticated
- Command authorization consults a RADIUS server to get authorization for commands entered by the user

Configuring Exec Authorization

When RADIUS exec authorization is performed, the Foundry device consults a RADIUS server to determine the privilege level of the authenticated user. To configure RADIUS exec authorization on the Foundry device, enter the following command:

```
BigIron(config)# aaa authorization exec default radius
```

Syntax: aaa authorization exec default radius | none

If you specify **none**, or omit the **aaa authorization exec** command from the device's configuration, no exec authorization is performed.

NOTE: If the **aaa authorization exec default radius** command exists in the configuration, following successful authentication the device assigns the user the privilege level specified by the foundry-privilege-level attribute received from the RADIUS server. If the **aaa authorization exec default radius** command does not exist in the configuration, then the value in the foundry-privilege-level attribute is ignored, and the user is granted Super User access.

Also note that in order for the **aaa authorization exec default radius** command to work, either the **aaa authentication enable default radius** command, or the **aaa authentication login privilege-mode** command must also exist in the configuration.

Configuring Command Authorization

When RADIUS command authorization is enabled, the Foundry device consults the list of commands supplied by the RADIUS server during authentication to determine whether a user can execute a command he or she has entered.

You enable RADIUS command authorization by specifying a privilege level whose commands require authorization. For example, to configure the Foundry device to perform authorization for the commands available at the Super User privilege level (that is; all commands on the device), enter the following command:

```
BigIron(config)# aaa authorization commands 0 default radius
```

Syntax: aaa authorization commands <privilege-level> default radius | tacacs+ | none

The <privilege-level> parameter can be one of the following:

- **0** – Authorization is performed (that is, the Foundry device looks at the command list) for commands available at the Super User level (all commands)
- **4** – Authorization is performed for commands available at the Port Configuration level (port-config and read-

only commands)

- **5** – Authorization is performed for commands available at the Read Only level (read-only commands)

NOTE: RADIUS command authorization can be performed only for commands entered from Telnet or SSH sessions, or from the console. No authorization is performed for commands entered at the Web management interface or IronView Network Manager.

NOTE: Since RADIUS command authorization relies on the command list supplied by the RADIUS server during authentication, you cannot perform RADIUS authorization without RADIUS authentication.

Command Authorization and Accounting for Console Commands

The Foundry device supports command authorization and command accounting for CLI commands entered at the console. To configure the device to perform command authorization and command accounting for console commands, enter the following:

```
BigIron(config)# enable aaa console
```

Syntax: enable aaa console

CAUTION: If you have previously configured the device to perform command authorization using a RADIUS server, entering the **enable aaa console** command may prevent the execution of any subsequent commands entered on the console.

This happens because RADIUS command authorization requires a list of allowable commands from the RADIUS server. This list is obtained during RADIUS authentication. For console sessions, RADIUS authentication is performed only if you have configured Enable authentication and specified RADIUS as the authentication method (for example, with the **aaa authentication enable default radius** command). If RADIUS authentication is never performed, the list of allowable commands is never obtained from the RADIUS server. Consequently, there would be no allowable commands on the console.

Configuring RADIUS Accounting

Foundry devices support RADIUS accounting for recording information about user activity and system events. When you configure RADIUS accounting on a Foundry device, information is sent to a RADIUS accounting server when specified events occur, such as when a user logs into the device or the system is rebooted.

Configuring RADIUS Accounting for Telnet/SSH (Shell) Access

To send an Accounting Start packet to the RADIUS accounting server when an authenticated user establishes a Telnet or SSH session on the Foundry device, and an Accounting Stop packet when the user logs out:

```
BigIron(config)# aaa accounting exec default start-stop radius
```

Syntax: aaa accounting exec default start-stop radius | tacacs+ | none

Configuring RADIUS Accounting for CLI Commands

You can configure RADIUS accounting for CLI commands by specifying a privilege level whose commands require accounting. For example, to configure the Foundry device to perform RADIUS accounting for the commands available at the Super User privilege level (that is; all commands on the device), enter the following command:

```
BigIron(config)# aaa accounting commands 0 default start-stop radius
```

An Accounting Start packet is sent to the RADIUS accounting server when a user enters a command, and an Accounting Stop packet is sent when the service provided by the command is completed.

NOTE: If authorization is enabled, and the command requires authorization, then authorization is performed before accounting takes place. If authorization fails for the command, no accounting takes place.

Syntax: aaa accounting commands <privilege-level> default start-stop radius | tacacs | none

The <privilege-level> parameter can be one of the following:

- **0** – Records commands available at the Super User level (all commands)
- **4** – Records commands available at the Port Configuration level (port-config and read-only commands)
- **5** – Records commands available at the Read Only level (read-only commands)

Configuring RADIUS Accounting for System Events

You can configure RADIUS accounting to record when system events occur on the Foundry device. System events include rebooting and when changes to the active configuration are made.

The following command causes an Accounting Start packet to be sent to the RADIUS accounting server when a system event occurs, and a Accounting Stop packet to be sent when the system event is completed:

```
BigIron(config)# aaa accounting system default start-stop radius
```

Syntax: aaa accounting system default start-stop radius | tacacs+ | none

Configuring an Interface as the Source for All RADIUS Packets

You can designate the lowest-numbered IP address configured an Ethernet port, POS port, loopback interface, or virtual interface as the source IP address for all RADIUS packets from the Layer 3 Switch. Identifying a single source IP address for RADIUS packets provides the following benefits:

- If your RADIUS server is configured to accept packets only from specific links or IP addresses, you can use this feature to simplify configuration of the RADIUS server by configuring the Foundry device to always send the RADIUS packets from the same link or source address.
- If you specify a loopback interface as the single source for RADIUS packets, RADIUS servers can receive the packets regardless of the states of individual links. Thus, if a link to the RADIUS server becomes unavailable but the client or server can be reached through another link, the client or server still receives the packets, and the packets still have the source IP address of the loopback interface.

The software contains separate CLI commands for specifying the source interface for Telnet, TACACS/TACACS+, and RADIUS packets. You can configure a source interface for one or more of these types of packets.

To specify an Ethernet or POS port or a loopback or virtual interface as the source for all RADIUS packets from the device, use the following CLI method. The software uses the lowest-numbered IP address configured on the port or interface as the source IP address for RADIUS packets originated by the device.

To specify the lowest-numbered IP address configured on a virtual interface as the device’s source for all RADIUS packets, enter commands such as the following:

```
BigIron(config)# int ve 1
BigIron(config-vif-1)# ip address 10.0.0.3/24
BigIron(config-vif-1)# exit
BigIron(config)# ip radius source-interface ve 1
```

The commands in this example configure virtual interface 1, assign IP address 10.0.0.3/24 to the interface, then designate the interface as the source for all RADIUS packets from the Layer 3 Switch.

Syntax: ip radius source-interface ethernet <portnum> | pos <portnum> | loopback <num> | ve <num>

The <num> parameter is a loopback interface or virtual interface number. If you specify an Ethernet or POS port, the <portnum> is the port’s number (including the slot number, if you are configuring a device).

Displaying RADIUS Configuration Information

The **show aaa** command displays information about all TACACS/TACACS+ and RADIUS servers identified on the device. For example:

```
BigIron# show aaa
Tacacs+ key: foundry
Tacacs+ retries: 1
Tacacs+ timeout: 15 seconds
Tacacs+ dead-time: 3 minutes
Tacacs+ Server: 207.95.6.90 Port:49:
                opens=6 closes=3 timeouts=3 errors=0
                packets in=4 packets out=4
no connection

Radius key: networks
Radius retries: 3
Radius timeout: 3 seconds
Radius dead-time: 3 minutes
Radius Server:  207.95.6.90 Auth Port=1645 Acct Port=1646:
                opens=2 closes=1 timeouts=1 errors=0
                packets in=1 packets out=4
no connection
```

The following table describes the RADIUS information displayed by the **show aaa** command.

Table 2.7: Output of the show aaa command for RADIUS

Field	Description
Radius key	The setting configured with the **radius-server key** command. At the Super User privilege level, the actual text of the key is displayed. At the other privilege levels, a string of periods (....) is displayed instead of the text.
Radius retries	The setting configured with the **radius-server retransmit** command.
Radius timeout	The setting configured with the **radius-server timeout** command.
Radius dead-time	The setting configured with the **radius-server dead-time** command.
Radius Server	For each RADIUS server, the IP address, and the following statistics are displayed: Auth Port — RADIUS authentication port number (default 1645) Acct Port — RADIUS accounting port number (default 1646) opens — Number of times the port was opened for communication with the server closes — Number of times the port was closed normally timeouts — Number of times port was closed due to a timeout errors — Number of times an error occurred while opening the port packets in — Number of packets received from the server packets out — Number of packets sent to the server
connection	The current connection status. This can be “no connection” or “connection active”.

The **show web** command displays the privilege level of Web management interface users. For example:

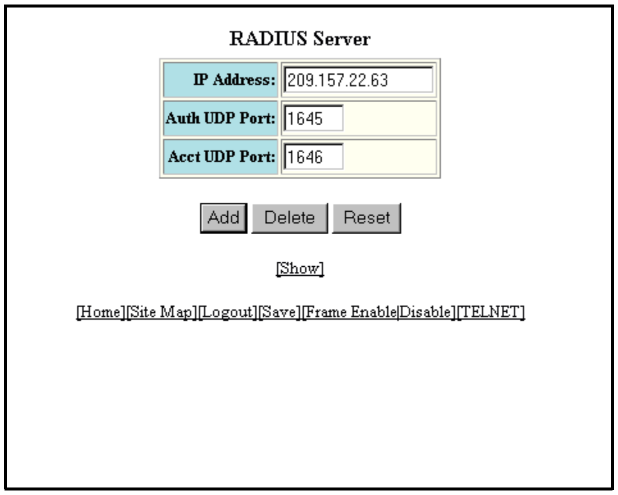
```
BigIron(config)# show web
User                              Privilege        IP address
set                                       0        192.168.1.234
```

Syntax: show web

USING THE WEB MANAGEMENT INTERFACE

To configure RADIUS using the Web management interface:

1. Log on to the device using a valid user name and password for read-write access. The System configuration panel is displayed.
2. If you configuring RADIUS authentication for Telnet access to the CLI, go to step 3. Otherwise, go to step 7.
3. Select the Management link to display the Management configuration panel.
4. Select Enable next to Telnet Authentication. You must enable Telnet authentication if you want to use TACACS/TACACS+ or RADIUS to authenticate Telnet access to the device.
5. Click Apply to apply the change.
6. Select the Home link to return to the System configuration panel.
7. Select the RADIUS link from the System configuration panel to display the RADIUS panel.
8. Change the retransmit interval, time out, and dead time if needed.
9. Enter the authentication key if applicable.
10. Click Apply if you changed any RADIUS parameters.
11. Select the RADIUS Server link.
    - If any RADIUS servers are already configured on the device, the servers are listed in a table. Select the Add RADIUS Server link to display the following panel.
    - If the device does not have any RADIUS servers configured, the following panel is displayed.



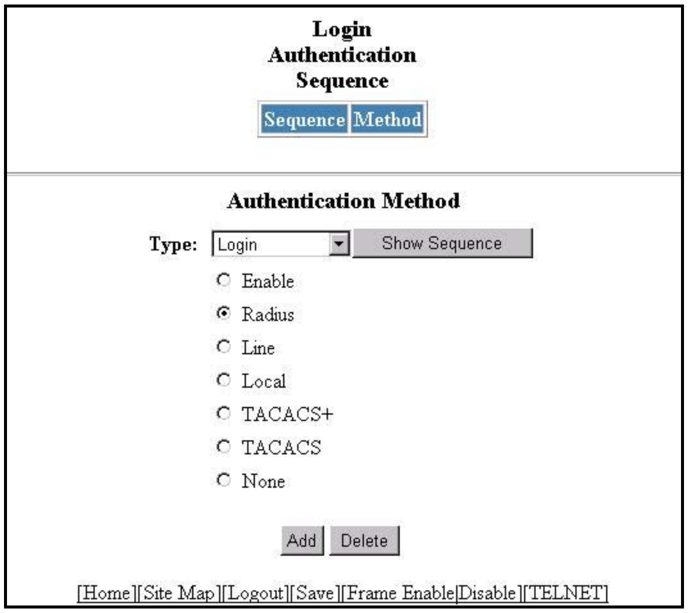
RADIUS Server

IP Address:	209.157.22.63
Auth UDP Port:	1645
Acct UDP Port:	1646

Add
Delete
Reset
[Show]
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

12. Enter the server's IP address in the IP Address field.
13. If needed, change the Authentication port and Accounting port. (The default values work in most networks.)

14. Click Home to return to the System configuration panel, then select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.
15. Select the Management link to display the Management configuration panel.
16. Select the Authentication Methods link to display the Login Authentication Sequence panel, as shown in the following example.

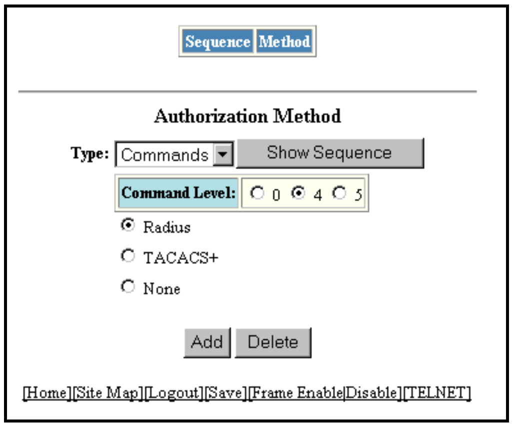


17. Select the type of access for which you are defining the authentication method list from the Type field's pulldown menu. Each type of access must have a separate authentication-method list. For example, to define the authentication-method list for logging into the CLI, select Login.
18. Select the primary authentication method by clicking on the radio button next to the method. For example, to use a RADIUS server as the primary means of authentication for logging on to the CLI, select RADIUS.
19. Click the Add button to save the change to the device's running-config file.

    The access type and authentication method you selected are displayed in the table at the top of the dialog. Each time you add an authentication method for a given access type, the software assigns a sequence number to the entry. When the user tries to log in using the access type you selected, the software tries the authentication sources in ascending sequence order until the access request is either approved or denied. Each time you add an entry for a given access type, the software increments the sequence number. Thus, if you want to use multiple authentication methods, make sure you enter the primary authentication method first, the secondary authentication method second, and so on.

    If you need to delete an entry, select the access type and authentication method for the entry, then click Delete.
20. Click Home to return to the System configuration panel, then select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

21. To configure RADIUS command authorization, select the Management link to display the Management configuration panel and select the Authorization Methods link to display the Authorization Method panel, as shown in the following example.



22. Select Commands from the Type field's pulldown menu.
23. Select a privilege level by clicking on one of the following radio buttons:
    - **0** – Authorization is performed for commands available at the Super User level (all commands)
    - **4** – Authorization is performed for commands available at the Port Configuration level (port-config and read-only commands)
    - **5** – Authorization is performed for commands available at the Read Only level (read-only commands)

NOTE: RADIUS authorization can be performed only for commands entered from Telnet or SSH sessions, or from the console. No authorization is performed for commands entered at the Web management interface or IronView Network Manager.

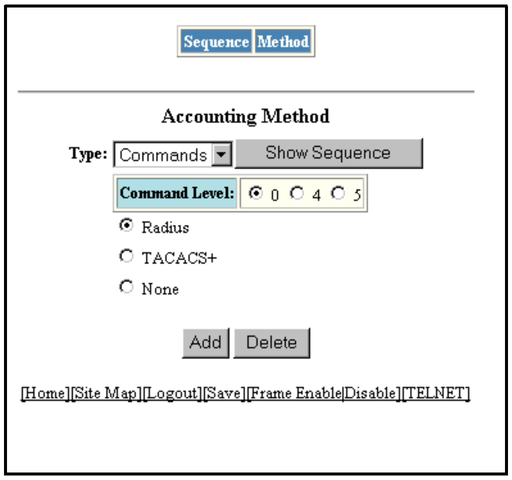
NOTE: Since RADIUS authorization relies on the command list supplied by the RADIUS server during authentication, you cannot perform RADIUS authorization without RADIUS authentication.

24. Click on the radio button next to Radius.
25. Click the Add button to save the change to the device's running-config file.

    The authorization method you selected are displayed in the table at the top of the dialog. Each time you add an authorization method for a given access type, the software assigns a sequence number to the entry. When authorization is performed, the software tries the authorization sources in ascending sequence order until the request is either approved or denied. Each time you add an entry for a given access type, the software increments the sequence number. Thus, if you want to use multiple authentication methods, make sure you enter the primary authentication method first, the secondary authentication method second, and so on.

    If you need to delete an entry, select the access type and authorization method for the entry, then click Delete.

26. To configure RADIUS accounting, select the Management link to display the Management configuration panel and select the Accounting Methods link to display the Accounting Method panel, as shown in the following example.



27. To send an Accounting Start packet to the RADIUS accounting server when an authenticated user establishes a Telnet or SSH session on the Foundry device, and an Accounting Stop packet when the user logs out, select Exec from the Type field's pulldown menu.
28. To configure RADIUS accounting for CLI commands, select Commands from the Type field's pulldown menu and select a privilege level by clicking on one of the following radio buttons:
    - **0** – Records commands available at the Super User level (all commands)
    - **4** – Records commands available at the Port Configuration level (port-config and read-only commands)
    - **5** – Records commands available at the Read Only level (read-only commands)
29. To configure RADIUS accounting to record when system events occur on the Foundry device, select System from the Type field's pulldown menu.
30. Click on the radio button next to Radius.
31. Click the Add button to save the change to the device's running-config file.

    The accounting method you selected are displayed in the table at the top of the dialog. Each time you add an accounting method for a given access type, the software assigns a sequence number to the entry. When accounting is performed, the software tries the accounting sources in ascending sequence order until the request is either approved or denied. Each time you add an entry for a given access type, the software increments the sequence number. Thus, if you want to use multiple accounting methods, make sure you enter the primary accounting method first, the secondary accounting method second, and so on.

    If you need to delete an entry, select the access type and accounting method for the entry, then click Delete.
32. Select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

Configuring Authentication-Method Lists

To implement one or more authentication methods for securing access to the device, you configure authentication-method lists that set the order in which the authentication methods are consulted.

In an authentication-method list, you specify the access method (Telnet, Web, SNMP, and so on) and the order in which the device tries one or more of the following authentication methods:

- Local Telnet login password

- Local password for the Super User privilege level
- Local user accounts configured on the device
- Database on a TACACS or TACACS+ server
- Database on a RADIUS server
- No authentication

NOTE: The TACACS/TACACS+, RADIUS, and Telnet login password authentication methods are not supported for SNMP access.

NOTE: To authenticate Telnet access to the CLI, you also must enable the authentication by entering the **enable telnet authentication** command at the global CONFIG level of the CLI. You cannot enable Telnet authentication using the Web management interface.

NOTE: You do not need an authentication-method list to secure access based on ACLs or a list of IP addresses. See “Using ACLs to Restrict Remote Access” on page 2-4 or “Restricting Remote Access to the Device to Specific IP Addresses” on page 2-8.

In an authentication-method list for a particular access method, you can specify up to seven authentication methods. If the first authentication method is successful, the software grants access and stops the authentication process. If the access is rejected by the first authentication method, the software denies access and stops checking.

However, if an error occurs with an authentication method, the software tries the next method on the list, and so on. For example, if the first authentication method is the RADIUS server, but the link to the server is down, the software will try the next authentication method in the list.

NOTE: If an authentication method is working properly and the password (and user name, if applicable) is not known to that method, this is not an error. The authentication attempt stops, and the user is denied access.

The software will continue this process until either the authentication method is passed or the software reaches the end of the method list. If the Super User level password is not rejected after all the access methods in the list have been tried, access is granted.

NOTE: In NetIron Chassis releases prior to 09.1.01, during local authentication, users who do not match the list of local users configured on a Foundry device are denied access to the device, whether or not other authentication methods are in the authentication methods list.

Starting with NetIron Chassis release 09.1.01, if a user cannot be authenticated using local authentication, then the next method on the authentication methods list is used to try to authenticate the user. If there is no method following local authentication, then the user is denied access to the device.

Configuration Considerations for Authentication-Method Lists

- For CLI access, you must configure authentication-method lists if you want the device to authenticate access using local user accounts or a RADIUS server. Otherwise, the device will authenticate using only the locally based password for the Super User privilege level.
- When no authentication-method list is configured specifically for Web management access, the device performs authentication using the SNMP community strings:
    - For read-only access, you can use the user name “get” and the password “public”. The default read-only community string is “public”.

    - Beginning with software release 05.1.00, there is no default read-write community string. Thus, by default, you cannot open a read-write management session using the Web management interface. You first must configure a read-write community string using the CLI. Then you can log on using "set" as the user name and the read-write community string you configure as the password. See "Configuring TACACS/TACACS+ Security" on page 2-23.

- If you configure an authentication-method list for Web management access and specify "local" as the primary authentication method, users who attempt to access the device using the Web management interface must supply a user name and password configured in one of the local user accounts on the device. The user ***cannot*** access the device by entering "set" or "get" and the corresponding SNMP community string.

- For devices that can be managed using IronView Network Manager, the default authentication method (if no authentication-method list is configured for SNMP) is the CLI Super User level password. If no Super User level password is configured, then access through IronView Network Manager is not authenticated. To use local user accounts to authenticate access through IronView Network Manager, configure an authentication-method list for SNMP access and specify "local" as the primary authentication method.

Examples of Authentication-Method Lists

Example 1: The following example shows how to configure authentication-method lists for the Web management interface, IronView Network Manager, and the Privileged EXEC and CONFIG levels of the CLI. In this example, the primary authentication method for each is "local". The device will authenticate access attempts using the locally configured user names and passwords first.

To configure an authentication-method list for the Web management interface, enter a command such as the following:

```
BigIron(config)# aaa authentication web-server default local
```

This command configures the device to use the local user accounts to authenticate access to the device through the Web management interface. If the device does not have a user account that matches the user name and password entered by the user, the user is not granted access.

To configure an authentication-method list for IronView Network Manager, enter a command such as the following:

```
BigIron(config)# aaa authentication snmp-server default local
```

This command configures the device to use the local user accounts to authenticate access attempts through any network management software, such as IronView Network Manager.

To configure an authentication-method list for the Privileged EXEC and CONFIG levels of the CLI, enter the following command:

```
BigIron(config)# aaa authentication enable default local
```

This command configures the device to use the local user accounts to authenticate attempts to access the Privileged EXEC and CONFIG levels of the CLI.

Example 2: To configure the device to consult a RADIUS server first to authenticate attempts to access the Privileged EXEC and CONFIG levels of the CLI, then consult the local user accounts if the RADIUS server is unavailable, enter the following command:

```
BigIron(config)# aaa authentication enable default radius local
```

Syntax: [no] aaa authentication snmp-server | web-server | enable | login default <method1> [<method2>] [<method3>] [<method4>] [<method5>] [<method6>] [<method7>]

The **snmp-server | web-server | enable | login** parameter specifies the type of access this authentication-method list controls. You can configure one authentication-method list for each type of access.

NOTE: TACACS/TACACS+ and RADIUS are supported only with the **enable** and **login** parameters.

The <method1> parameter specifies the primary authentication method. The remaining optional <method> parameters specify additional methods to try if an error occurs with the primary method. A method can be one of the values listed in the Method Parameter column in the following table.

Table 2.8: Authentication Method Values

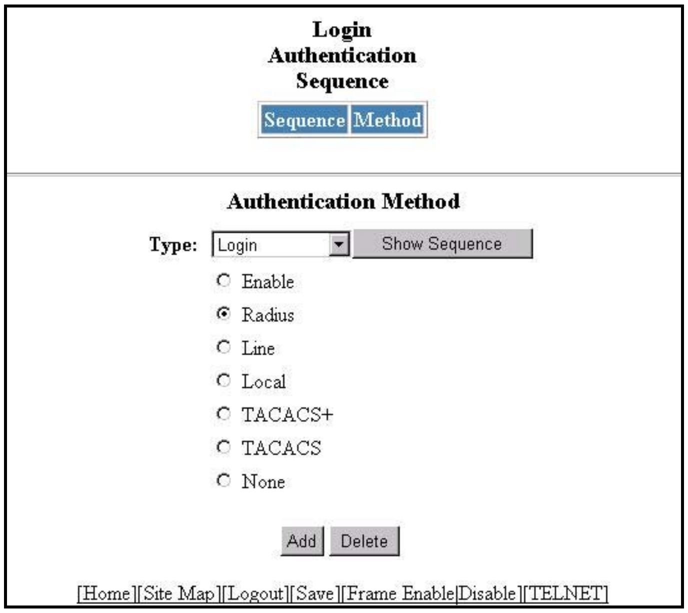
Method Parameter	Description
line	Authenticate using the password you configured for Telnet access. The Telnet password is configured using the **enable telnet password…** command. See “Setting a Telnet Password” on page 2-15.
enable	Authenticate using the password you configured for the Super User privilege level. This password is configured using the **enable super-user-password…** command. See “Setting Passwords for Management Privilege Levels” on page 2-16.
local	Authenticate using a local user name and password you configured on the device. Local user names and passwords are configured using the **username…** command. See “Configuring a Local User Account” on page 2-19.
tacacs	Authenticate using the database on a TACACS server. You also must identify the server to the device using the **tacacs-server** command.
tacacs+	Authenticate using the database on a TACACS+ server. You also must identify the server to the device using the **tacacs-server** command.
radius	Authenticate using the database on a RADIUS server. You also must identify the server to the device using the **radius-server** command. See “Configuring RADIUS Security” on page 2-40.
none	Do not use any authentication method. The device automatically permits access.

USING THE WEB MANAGEMENT INTERFACE

To configure an authentication-method list with the Web management interface, use the following procedure. This example to causes the device to use a RADIUS server to authenticate attempts to log in through the CLI:

1. Log on to the device using a valid user name and password for read-write access. The System configuration panel is displayed.
2. Select the Management link to display the Management configuration panel.

3. Select the Authentication Methods link to display the Login Authentication Sequence panel, as shown in the following example.


Login Authentication Sequence
Sequence
Method

Authentication Method

Type: Login

- Enable
- [✓ Radius]
- Line
- Local
- TACACS+
- TACACS
- None

Add
Delete
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

4. Select the type of access for which you are defining the authentication method list from the Type field's pulldown menu. Each type of access must have a separate authentication-method list. For example, to define the authentication-method list for logging into the CLI, select Login.

5. Select the primary authentication method by clicking the button next to the method. For example, to use a RADIUS server as the primary means of authentication for logging on to the CLI, select RADIUS.

6. Click the Add button to save the change to the device's running-config file. The access type and authentication method you selected are displayed in the table at the top of the dialog. Each time you add an authentication method for a given access type, the software assigns a sequence number to the entry. When the user tries to log in using the access type you selected, the software tries the authentication sources in ascending sequence order until the access request is either approved or denied. Each time you add an entry for a given access type, the software increments the sequence number. Thus, if you want to use multiple authentication methods, make sure you enter the primary authentication method first, the secondary authentication method second, and so on.

   If you need to delete an entry, select the access type and authentication method for the entry, then click Delete.

7. Select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

Using the Configuration Bootguard Feature

It may be possible for an unauthorized user to boot the Foundry device using the default-config (that is, the configuration used for new devices that do not have a startup-config file) and then use the Foundry device to gain access to other devices in the network. To prevent this from happening, you can enable the ***configuration bootguard*** feature.

When enabled, the configuration bootguard feature causes the Foundry device to erase its startup-config file in the event that the device is booted using the default-config, and a Super User password had previously been stored on the device. As a result of erasing its startup-config file, the Foundry device would lose connectivity within the network, but it would prevent an unauthorized user from using the Foundry device to gain access to other devices in the network, in addition to preventing the unauthorized user from obtaining the contents of the startup-config file.

How the Configuration Bootguard Feature Works

Starting in Enterprise software release 08.0.00, whenever you save the running-config to flash memory (with the **write memory** command), the Foundry device sets an internal flag related to the configuration bootguard feature:

- If the configuration bootguard feature is enabled, and a Super User password exists in the saved configuration, then the flag is set to ON.
- If the configuration bootguard feature is not enabled, or there is no Super User password configured on the device, then the flag is set to OFF.

The setting for this flag is saved in non-volatile memory, so it is saved across software reloads. When the Foundry device is booted, it checks the value of the configuration bootguard flag. Depending on the value of the flag, the Foundry device does one of the following:

- If the flag is ON, and the device is being booted with the default configuration, then the startup-config file in flash memory is erased.
- If the flag is ON, and the device is being booted with a configuration other than the default configuration, then the device boots normally.
- If the flag is OFF, then the device boots normally.

Enabling and Disabling the Configuration Bootguard Feature

To enable the configuration bootguard feature, enter the following commands:

```
BigIron(config)# enable-bootguard
BigIron(config)# write memory
```

To disable the configuration bootguard feature, enter the following commands:

```
BigIron(config)# no enable-bootguard
BigIron(config)# write memory
```

Syntax: [no] enable-bootguard

The configuration bootguard feature is disabled by default.

Chapter 3
Configuring Secure Shell (Non-Terathon Devices)

Overview

Secure Shell (SSH) is a mechanism for allowing secure remote access to management functions on a Foundry device. SSH provides a function similar to Telnet. Users can log into and configure the device using a publicly or commercially available SSH client program, just as they can with Telnet. However, unlike Telnet, which provides no security, SSH provides a secure, encrypted connection to the device.

SSH supports Arcfour, IDEA, Blowfish, DES (56-bit) and Triple DES (168-bit) data encryption methods. Nine levels of data compression are available. You can configure your SSH client to use any one of these data compression levels when connecting to a Foundry device.

Foundry devices also support Secure Copy (SCP) for securely transferring files between a Foundry device and SCP-enabled remote hosts. See “Using Secure Copy” on page 3-12 for more information.

NOTE: SSH is supported on the following Foundry devices:

- NetIron Internet backbone routers
- BigIron Chassis devices with Management II or higher modules
- FastIron II and FastIron II Plus (switch and basic Layer 3 code only)
- NetIron Layer 3 Switch (stackable, octal version)
- ServerIron XL stackable and ServerIron Chassis devices
- FastIron Edge Switch series
- FastIron Workgroup Layer 2 Switch (8MB models only, switch code only, releases prior to 07.1.26d only)

SSH Version 2 Support

SSH Version 2 is supported on Foundry devices running Enterprise software release 07.8.00 and later.

SSHv2 is a substantial revision of Secure Shell, comprising the following hybrid protocols and definitions:

- SSH Transport Layer Protocol
- SSH Authentication Protocol
- SSH Connection Protocol
- GSSAPI Authentication and Key Exchange for the Secure Shell Protocol
- Generic Message Exchange Authentication For SSH

- SECSH Public Key File Format
- SSH Fingerprint Format
- SSH Protocol Assigned Numbers
- DNS to Securely Publish SSH Key Fingerprints (not support in release 02.1.00 for the NetIron IMR 640 and release 02.1.00 for the BigIron MG8 and NetIron 40G)
- SSH Transport Layer Encryption Modes
- Session Channel Break Extension
- SCP/SFTP/SSH URI Format (only SCP is supported in release 02.1.00 for the NetIron IMR 640 and release 02.1.00 for the BigIron MG8 and NetIron 40G)

NOTE: The CLI commands for setting up and configuring SSHv2 on a Foundry device are identical to those for SSHv1.

If you are using redundant management modules, you can synchronize the RSA host key pair between the active and standby modules by entering the **sync-standby code** command at the Privileged EXEC level of the CLI. When you subsequently enter the **write memory** command, the RSA host key pair is synchronized to the standby module.

Foundry's SSHv2 implementation is compatible with all versions of the SSHv2 protocol (2.1, 2.2, and so on). At the beginning of an SSH session, the Foundry device negotiates the version of SSHv2 to be used. The highest version of SSHv2 supported by both the Foundry device and the client is the version that is used for the session. Once the SSHv2 version is negotiated, the encryption algorithm with the highest security ranking is selected to be used for the session.

Tested SSHv2 Clients

The following SSH clients have been tested with SSHv2:

- SSH Secure Shell 3.2.3
- Van Dyke SecureCRT 4.0
- F-Secure SSH Client 5.3
- Tera Term Pro 3.1.3
- PuTTY 0.54
- OpenSSH 3.5_p1

Supported Encryption Algorithms for SSHv2

The following encryption algorithms are supported with Foundry implementation of SSHv2:

- AES
- Twofish
- Blowfish
- 3DES
- Arcfour(RC4)
- CAST
- None selected

Supported MAC (Message Authentication Code) Algorithms

The following MAC algorithms are supported with Foundry implementation of SSHv2:

- MD5
- SHA

- None selected

Configuring SSH

Foundry's implementation of SSH supports two kinds of user authentication:

- ***RSA challenge-response authentication***, where a collection of public keys are stored on the device. Only clients with a private key that corresponds to one of the stored public keys can gain access to the device using SSH.
- ***Password authentication***, where users attempting to gain access to the device using an SSH client are authenticated with passwords stored on the device or on a TACACS/TACACS+ or RADIUS server

Both kinds of user authentication are enabled by default. You can configure the device to use one or both of them.

To configure Secure Shell on a Foundry device, do the following:

1. Set the Foundry device's host name and domain name.
2. Generate a host RSA public and private key pair for the device.
3. Configure RSA challenge-response authentication.
4. Set optional parameters.

You can also view information about active SSH connections on the device as well as terminate them.

Setting the Host Name and Domain Name

If you have not already done so, establish a host name and domain name for the Foundry device. For example:

```
BigIron(config)# hostname BigIron
BigIron(config)# ip dns domain-name foundrynet.com
```

Syntax: hostname <name>

Syntax: ip dns domain-name <name>

Generating a Host I Key Pair

When SSH is configured, a public and private ***host RSA key pair*** is generated for the Foundry device. The SSH server on the Foundry device uses this host RSA key pair, along with a dynamically generated ***server RSA key pair***, to negotiate a session key and encryption method with the client trying to connect to it.

The host RSA key pair is stored in the Foundry device's system-config file. Only the public key is readable. The public key should be added to a "known hosts" file (for example, $HOME/.ssh/known_hosts on UNIX systems) on the clients who want to access the device. Some SSH client programs add the public key to the known hosts file automatically; in other cases, you must manually create a known hosts file and place the Foundry device's public key in it. See "Providing the Public Key to Clients" on page 3-5 for an example of what to place in the known hosts file.

To generate a public and private RSA host key pair for the most Foundry devices, enter the following commands:

```
BigIron(config)# crypto key generate rsa
BigIron(config)# write memory
```

The **crypto key generate [rsa]** command places an RSA host key pair in the running-config file and enables SSH on the device.

To disable SSH, you must delete the RSA host key pair. To do this in SSHv1, enter the following commands:

```
BigIron(config)# crypto key zeroize rsa
BigIron(config)# write memory
```

The **crypto key zeroize [rsa]** command deletes the RSA host key pair in the running-config file and disables SSH on the device.

Syntax: crypto key generate | zeroize rsa

You can optionally configure the Foundry device to hide the RSA host key pair in the running-config file. To do this, enter the following command:

```
BigIron# ssh no-show-host-keys
```

Syntax: ssh no-show-host-keys

After entering the **ssh no-show-host-keys** command, you can display the RSA host key pair in the running-config file with the following command:

```
BigIron# ssh show-host-keys
```

Syntax: ssh show-host-keys

Notes:

- If the Foundry device does not have internal memory (for example, BxGCM), the RSA host key pair is stored in the startup-config file. In this case, the **ssh show-host-keys** command allows the RSA host key pair to be viewed, and the **ssh no-show-host-keys** command prevents the RSA host key pair from being viewed.
- If an RSA host key pair is stored in internal memory on the Foundry device, it is used even if the startup-config file contains a different RSA host key pair.
- If no RSA host key pair is stored in internal memory, but the startup-config file contains an RSA host key pair, the key pair in the startup-config file is used. If you later generate an RSA host key pair with the **crypto key generate** command, the new key pair takes effect only after you store it in internal memory with the **write memory** command and reboot the Foundry device.
- If no RSA host key pair is stored in internal memory, and the startup-config file contains an RSA host key pair, the first time you enter the **write memory** command, it will save the RSA host key pair in the startup-config file to internal memory and remove it from the startup-config file.
- If no RSA host key pair is stored in internal memory, the startup-config file contains an RSA host key pair, and you generate an RSA host key pair with the **crypto key generate** command, the new pair is stored in internal memory the first time you enter the **write memory** command.
- The **crypto key zeroize** command disables the currently active RSA host key pair. If you subsequently enter the **write memory** command without generating another RSA host key pair, the RSA host key pair stored in internal memory is removed.
- On devices managed by the VM1, if you erase the startup-config file, the RSA host key pair will still reside in internal memory. To remove the RSA host key pair from internal memory, you must enter the **crypto key zeroize** command.
- If you enter the **ssh no-show-host-keys** command to hide the RSA host key pair in the running-config file, then reload the software, the RSA host key pair is once again visible in the running-config file. The setting to hide the RSA host key pair is not carried across software reloads.
- In a configuration using redundant management modules, if the active module has an RSA host key pair, but the standby module does not, the RSA host key pair is not carried over when switchover occurs. You must create an RSA host key pair on the standby module manually.
- The SSH key generation process causes UDLD-enabled interfaces to go down instantaneously. This in turn requires the reconvergence of the route tables on the routers across the network. Non-UDLD-enabled interfaces do not experience this issue.

Providing the Public Key to Clients

If you are using SSH to connect to a Foundry device from a UNIX system, you may need to add the Foundry device's public key to a "known hosts" file; for example, $HOME/.ssh/known_hosts. The following is an example of an entry in a known hosts file:

```
10.10.20.10 1024 37 118771881862677030464851288737258046856031640635887679230111
84247022636175804896633384620574930068397650231698985431857279323745963240790218
03229084221453472515782437007702806627934784079949643404159653290224014833380339
09542147367974638560060162945329307563502804231039654388220432832662804242569361
58342816331
```

In this example, 10.10.20.10 is the IP address of an SSH-enabled Foundry Layer 2 Switch or Layer 3 Switch. The second number, 1024, is the size of the host key, and the third number, 37, is the encoded public exponent. The remaining text is the encoded modulus.

Configuring RSA Challenge-Response Authentication

With RSA challenge-response authentication, a collection of clients' public keys are stored on the Foundry device. Clients are authenticated using these stored public keys. Only clients that have a private key that corresponds to one of the stored public keys can gain access to the device using SSH.

When RSA challenge-response authentication is enabled, the following events occur when a client attempts to gain access to the device using SSH:

1. The client sends its public key to the Foundry device.
2. The Foundry device compares the client's public key to those stored in memory.
3. If there is a match, the Foundry device uses the public key to encrypt a random sequence of bytes.
4. The Foundry device sends these encrypted bytes to the client.
5. The client uses its private key to decrypt the bytes.
6. The client sends the decrypted bytes back to the Foundry device.
7. The Foundry device compares the decrypted bytes to the original bytes it sent to the client. If the two sets of bytes match, it means that the client's private key corresponds to an authorized public key, and the client is authenticated.

Setting up RSA challenge-response authentication consists of the following steps:

8. Importing authorized public keys into the Foundry device.
9. Enabling RSA challenge response authentication

Importing Authorized Public Keys into the Foundry Device

SSH clients that support RSA authentication normally provide a utility to generate an RSA key pair. The private key is usually stored in a password-protected file on the local host; the public key is stored in another file and is not protected. You should collect one public key from each client to be granted access to the Foundry device and place all of these keys into one file. This public key file is imported into the Foundry device.

The following is an example of a public key file containing two public keys:

```
1024 65537 162566050678380006149460550286514061230306797782065166110686648548574
94957339232259963157379681924847634614532742178652767231995746941441604714682680
00644536790333304202912490569077182886541839656556769025432881477252978135927821
67540629478392662275128774861815448523997023618173312328476660721888873946758201
 user@csp_client
1024 35 152676199889856769693556155614587291553826312328095300428421494164360924
76207475545234679268443233762295312979418833525975695775705101805212541008074877
26586119857422702897004112168852145074087969840642408451742714558592361693705908
74837875599405503479603024287131312793895007927438074972787423695977635251943 ro
ot@unix_machine
```

You can import the authorized public keys into the active configuration by loading them from a file on a TFTP server or from a file on the Management IV module's PCMCIA flash card. Once the authorized public keys are loaded, you can optionally save them to the startup-config file. If you import a public key file from a TFTP server or PCMCIA flash card, the file is automatically loaded into the active configuration the next time the device is booted.

To load the public key file onto the Management IV module's PCMCIA flash card, you can copy it from a TFTP server or from a Secure Copy (SCP)-enabled client.

For example, to copy a public key file called pkeys.txt from a TFTP server to a PCMCIA flash card in slot 1:

```
BigIron# copy tftp slot1 192.168.1.234 pkeys.txt
```

Syntax: copy tftp slot1 | slot2 <tftp-server-ip-addr> <from-name> [<to-name>]

Foundry devices support Secure Copy (SCP) for securely transferring files between hosts on a network. Note that when you copy files using SCP, you enter the commands on the SCP-enabled client, rather than the console on the Foundry device.

For example, to copy a public key file called pkeys.txt from an SCP-enabled client to a PCMCIA flash card in slot 1 on a Management IV module, enter a command such as the following on the SCP-enabled client:

```
C:\> scp c:\pkeys.txt user@BigIron:a:/pkeys.txt
```

If password authentication is enabled for SSH, the user will be prompted for a password in order to copy the file. See "Using Secure Copy" on page 3-12 for more information on SCP.

After the file is loaded onto the TFTP server or PCMCIA flash card, it can be imported into the active configuration each time the device is booted.

To cause a public key file called pkeys.txt to be loaded from the Management IV module's PCMCIA flash card each time the Foundry device is booted, enter the following command:

```
BigIron(config)# ip ssh pub-key-file slot1 pkeys.txt
```

Syntax: ip ssh pub-key-file slot1 | slot2 <filename>

To cause a public key file called pkeys.txt to be loaded from a TFTP server each time the Foundry device is booted, enter a command such as the following:

```
BigIron(config)# ip ssh pub-key-file tftp 192.168.1.234 pkeys.txt
```

Syntax: ip ssh pub-key-file tftp <tftp-server-ip-addr> <filename>

The <tftp-server-ip-addr> variable is the IP address of the tftp server that contains the public key file that you want to import into the Foundry device.

The <filename> variable is the name of the dsa public key file that you want to import into the Foundry device.

To display the currently loaded public keys, enter the following command:

```
BigIron# show ip client-pub-key

1024 65537 162566050678380006149460550286514061230306797782065166110686648548574
949573392322599631573796819248476346145327421786527672319957469414416047146826 80
006445367903333042029124905690771828865418396565567690254328814772529781359278 21
675406294783926622751287748618154485239970236181733123284766607218888739467582 01
 user@csp_client

1024 35 152676199889856769693556155614587291553826312328095300428421494164360924
762074755452346792684432337622953129794188335259756957757051018052125410080748 77
265861198574227028970041121688521450740879698406424084517427145585923616937059 08
7483787559940550347960302428713131279389500792743807497278742369597763525194 3 ro
ot@unix_machine

There are 2 authorized client public keys configured
```

Syntax: show ip client-pub-key

To clear the public keys from the active configuration, enter the following command:

```
BigIron# clear public-key
```

Syntax: clear public-key

To reload the public keys from the file on the TFTP server or PCMCIA flash card, enter the following command:

```
BigIron(config)# ip ssh pub-key-file reload
```

Syntax: ip ssh pub-key-file reload

Once the public keys are part of the active configuration, you can make them part of the startup-config file. The startup-config file can contain a maximum of 10 public keys. If you want to store more than 10 public keys, keep them in a file on a TFTP server or PCMCIA flash card, where they will be loaded into the active configuration when the device is booted.

To make the public keys in the active configuration part of the startup-config file, enter the following commands:

```
BigIron(config)# ip ssh pub-key-file flash-memory
BigIron(config)# write memory
```

Syntax: ip ssh pub-key-file flash-memory

To clear the public keys from the startup-config file (if they are located there), enter the following commands:

```
BigIron# clear public-key
BigIron# write memory
```

Enabling RSA Challenge-Response Authentication

RSA challenge-response authentication is enabled by default. You can disable or re-enable it manually.

To enable RSA challenge-response authentication:

```
BigIron(config)# ip ssh rsa-authentication yes
```

To disable RSA challenge-response authentication:

```
BigIron(config)# ip ssh rsa-authentication no
```

Syntax: ip ssh rsa-authentication yes | no

Setting Optional Parameters

You can adjust the following SSH settings on the Foundry device:

- The number of SSH authentication retries
- The server RSA key size
- The user authentication method the Foundry device uses for SSH connections
- Whether the Foundry device allows users to log in without supplying a password
- The port number for SSH connections
- The SSH login timeout value
- A specific interface to be used as the source for all SSH traffic from the device
- The maximum idle time for SSH sessions

Setting the Number of SSH Authentication Retries

By default, the Foundry device attempts to negotiate a connection with the connecting host three times. The number of authentication retries can be changed to between 1 – 5.

For example, the following command changes the number of authentication retries to 5:

```
BigIron(config)# ip ssh authentication-retries 5
```

Syntax: ip ssh authentication-retries <number>

Setting the Server RSA Key Size

The default size of the dynamically generated server RSA key is 768 bits. The size of the server RSA key can be between 512 – 896 bits.

For example, the following command changes the server RSA key size to 896 bits:

```
BigIron(config)# ip ssh key-size 896
```

Syntax: ip ssh key-size <number>

NOTE: The **ip ssh key-size** command is not applicable to SSHv2 implementation.

NOTE: The size of the ***host*** RSA key that resides in the system-config file is always 1024 bits and cannot be changed.

Deactivating User Authentication

After the SSH server on the Foundry device negotiates a session key and encryption method with the connecting client, user authentication takes place. Foundry's implementation of SSH supports RSA challenge-response authentication and password authentication.

With RSA challenge-response authentication, a collection of clients' public keys are stored on the Foundry device. Clients are authenticated using these stored public keys. Only clients that have a private key that corresponds to one of the stored public keys can gain access to the device using SSH.

With password authentication, users are prompted for a password when they attempt to log into the device (provided empty password logins are not allowed; see "Enabling Empty Password Logins" on page 3-9). If there is no user account that matches the user name and password supplied by the user, the user is not granted access.

You can deactivate one or both user authentication methods for SSH. Note that deactivating both authentication methods essentially disables the SSH server entirely.

To disable RSA challenge-response authentication:

```
BigIron(config)# ip ssh rsa-authentication no
```

Syntax: ip ssh rsa-authentication no | yes

To deactivate password authentication:

```
BigIron(config)# ip ssh password-authentication no
```

Syntax: ip ssh password-authentication no | yes

Enabling Empty Password Logins

By default, empty password logins are not allowed. This means that users with an SSH client are always prompted for a password when they log into the device. To gain access to the device, each user must have a user name and password. Without a user name and password, a user is not granted access. See “Setting Up Local User Accounts” on page 2-19 for information on setting up user names and passwords on Foundry devices.

If you enable empty password logins, users are ***not*** prompted for a password when they log in. Any user with an SSH client can log in without being prompted for a password.

To enable empty password logins:

```
BigIron(config)# ip ssh permit-empty-passwd yes
```

Syntax: ip ssh permit-empty-passwd no | yes

Setting the SSH Port Number

By default, SSH traffic occurs on TCP port 22. You can change this port number. For example, the following command changes the SSH port number to 2200:

```
BigIron(config)# ip ssh port 2200
```

Note that if you change the default SSH port number, you must configure SSH clients to connect to the new port. Also, you should be careful not to assign SSH to a port that is used by another service. If you change the SSH port number, Foundry recommends that you change it to a port number greater than 1024.

Syntax: ip ssh port <number>

Setting the SSH Login Timeout Value

When the SSH server attempts to negotiate a session key and encryption method with a connecting client, it waits a maximum of 120 seconds for a response from the client. If there is no response from the client after 120 seconds, the SSH server disconnects. You can change this timeout value to between 1 – 120 seconds. For example, to change the timeout value to 60 seconds:

```
BigIron(config)# ip ssh timeout 60
```

Syntax: ip ssh timeout <seconds>

Designating an Interface as the Source for All SSH Packets

You can designate a loopback interface, virtual interface, POS port, or Ethernet port as the source for all SSH packets from the device. The software uses the IP address with the numerically lowest value configured on the port or interface as the source IP address for SSH packets originated by the device.

NOTE: When you specify a single SSH source, you can use only that source address to establish SSH management sessions with the Foundry device.

To specify the numerically lowest IP address configured on a loopback interface as the device’s source for all SSH packets, enter commands such as a the following:

```
BigIron(config)# int loopback 2
BigIron(config-lbif-2)# ip address 10.0.0.2/24
BigIron(config-lbif-2)# exit
BigIron(config)# ip ssh source-interface loopback 2
```

The commands in this example configure loopback interface 2, assign IP address 10.0.0.2/24 to the interface, then designate the interface as the source for all SSH packets from the Layer 3 Switch.

Syntax: ip ssh source-interface ethernet <portnum> | pos <portnum> | loopback <num> | ve <num>

The <num> parameter is a loopback interface or virtual interface number. If you specify an Ethernet or POS port, the <portnum> is the port’s number (including the slot number, if you are configuring a Chassis device). For example:

```
BigIron(config)# interface ethernet 1/4
BigIron(config-if-1/4)# ip address 209.157.22.110/24
BigIron(config-if-1/4)# exit
BigIron(config)# ip ssh source-interface ethernet 1/4
```

Configuring Maximum Idle Time for SSH Sessions

By default, SSH sessions do not time out. Optionally, you can set the amount of time an SSH session can be inactive before the Foundry device closes it. For example, to set the maximum idle time for SSH sessions to 30 minutes:

```
BigIron(config)# ip ssh idle-time 30
```

Syntax: ip ssh idle-time <minutes>

If an established SSH session has no activity for the specified number of minutes, the Foundry device closes it. An idle time of 0 minutes (the default value) means that SSH sessions never timeout. The maximum idle time for SSH sessions is 240 minutes.

Displaying SSH Connection Information

Up to five SSH connections can be active on the Foundry device. To display information about SSH connections, enter the following command:

```
BigIron# show ip ssh
Connection     Version     Encryption     State     Username
    1            1.5         ARCFOUR        0x82      neville
    2            1.5          IDEA          0x82      lynval
    3            1.5          3DES          0x82      terry
    4            1.5          none          0x00
    5              1.5          none           0x00
```

Syntax: show ip ssh

This display shows the following information about the active SSH connections:.

Table 3.1: SSH Connection Information

This Field...	Displays...
Connection	The SSH connection ID. This can be from 1 – 5.
Version	The SSH version number. This should always be 1.5.
Encryption	The encryption method used for the connection. This can be IDEA, ARCFOUR, DES, 3DES, or BLOWFISH.

Table 3.1: SSH Connection Information (Continued)

This Field...	Displays...
State	The connection state. This can be one of the following: 0x00 Server started to send version number to client. 0x01 Server sent version number to client. 0x02 Server received version number from client. 0x20 Server sent public key to client. 0x21 Server is waiting for client's session key. 0x22 Server received session key from client. 0x23 Server is verifying client's session key. 0x24 Client's session key is verified. 0x25 Server received client's name. 0x40 Server is authenticating client. 0x41 Server is continuing to authenticate client after one or more failed attempts. 0x80 Server main loop started after successful authentication. 0x81 Server main loop sent a message to client. 0x82 Server main loop received a message from client.
Username	The user name for the connection.

The **show who** command also displays information about SSH connections. For example:

```
BigIron#show who
Console connections:
 established, active
Telnet connections:
 1 closed
 2 closed
 3 closed
 4 closed
 5 closed
SSH connections:
 1 established, client ip address 209.157.22.8
 16 seconds in idle
 2 established, client ip address 209.157.22.21
 42 seconds in idle
 3 established, client ip address 209.157.22.68
 49 seconds in idle
 4 closed
 5 closed
```

Syntax: show who

To terminate one of the active SSH connections, enter the following command:

```
BigIron# kill ssh 1
```

Syntax: kill ssh <connection-id>

Sample SSH Configuration

The following is a sample SSH configuration for a Foundry device.

```
hostname BigIron
ip dns domain-name foundrynet.com
!
aaa authentication login default local
username neville password .....
username lynval password .....
username terry password .....
!
ip ssh permit-empty-passwd no
!
ip ssh pub-key-file tftp 192.168.1.234 pkeys.txt
!
crypto key generate rsa public_key "1024 35 144460146631716543532035011163035196
4119319512520589445263746240952227550502084508730298520996034623917299567632935 7
2477753018866626789819564825318155162468139452068167261082818831041396224230129 6
2688393717676977618498409310098401707536938707100663796665087722467797948680265 1
45832421805508331331394853490240 9 BigIron@foundrynet.com"
!
crypto key generate rsa private_key "**************************"
!
ip ssh authentication-retries 5
```

This **aaa authentication login default local** command configures the device to use the local user accounts to authenticate users attempting to log in.

Three user accounts are configured on the device. The **ip ssh permit-empty-passwd no** command causes users always to be prompted for a password when they attempt to establish an SSH connection. Since the device uses local user accounts for authentication, only these three users are allowed to connect to the device using SSH.

The **ip ssh pub-key-file tftp** command causes a public key file called pkeys.txt to be loaded from a TFTP server at 192.168.1.234. To gain access to the Foundry device using SSH, a user must have a private key that corresponds to one of the public keys in this file.

The **crypto key generate rsa public_key** and **crypto key generate rsa private_key** statements are both generated by the **crypto key generate rsa** command. By default, the RSA host key pair appears in the running-config file, but not in the startup-config file. You can optionally configure the Foundry device to hide the RSA host key pair in the running-config file with the **ssh no-show-host-keys** command. The actual private key is never visible in either the running-config file or the startup-config file.

You may need to copy the public key to a "known hosts" file (for example, $HOME/.ssh/known_hosts on UNIX systems) on the clients who want to access the device. See "Providing the Public Key to Clients" on page 3-5 for an example of what to place in the known hosts file.

The **ip ssh authentication-retries 5** command sets the number of times the Foundry device attempts to negotiate a connection with the connecting host to 5.

Using Secure Copy

Secure Copy (SCP) uses security built into SSH to transfer files between hosts on a network, providing a more secure file transfer method than Remote Copy (RCP) or FTP. SCP automatically uses the authentication methods, encryption algorithm, and data compression level configured for SSH. For example, if password authentication is enabled for SSH, the user is prompted for a user name and password before SCP allows a file to be transferred. No additional configuration is required for SCP on top of SSH.

You can use SCP to copy files on the Foundry device, including the startup-config and running-config files, to or from an SCP-enabled remote host.

SCP is enabled by default and can be disabled. To disable SCP, enter the following command:

```
BigIron(config)# ip ssh scp disable
```

Syntax: ip ssh scp disable | enable

NOTE: If you disable SSH, SCP is also disabled.

The following are examples of using SCP to transfer files from and to a Foundry device

NOTE: When using SCP, you enter the **scp** commands on the SCP-enabled client, rather than the console on the Foundry device.

NOTE: Certain SCP client options, including -p and -r, are ignored by the SCP server on the Foundry device. If an option is ignored, the client is notified.

To copy a configuration file (c:\cfg\foundry.cfg) to the running-config file on a Foundry device at 192.168.1.50 and log in as user terry, enter the following command on the SCP-enabled client:

```
C:\> scp c:\cfg\foundry.cfg terry@192.168.1.50:runConfig
```

If password authentication is enabled for SSH, the user is prompted for user terry's password before the file transfer takes place.

To copy the configuration file to the startup-config file:

```
C:\> scp c:\cfg\foundry.cfg terry@192.168.1.50:startConfig
```

To copy the configuration file to a file called config1.cfg on the PCMCIA flash card in slot 1 on a Management IV module:

```
C:\> scp c:\cfg\foundry.cfg terry@192.168.1.50:a:/config1.cfg
```

To copy the configuration file to a file called config1.cfg on the PCMCIA flash card in slot 2 on a Management IV module:

```
C:\> scp c:\cfg\foundry.cfg terry@192.168.1.50:b:/config1.cfg
```

To copy the running-config file on a Foundry device to a file called c:\cfg\fdryrun.cfg on the SCP-enabled client:

```
C:\> scp terry@192.168.1.50:runConfig c:\cfg\fdryrun.cfg
```

To copy the startup-config file on a Foundry device to a file called c:\cfg\fdrystart.cfg on the SCP-enabled client:

```
C:\> scp terry@192.168.1.50:startConfig c:\cfg\fdrystart.cfg
```

To copy a file called config1.cfg on the PCMCIA flash card in slot 1 on a Management IV module to the SCP-enabled client:

```
C:\> scp terry@192.168.1.50:a:/config1.cfg c:\cfg\config1.cfg
```

To copy a file called config2.cfg on the PCMCIA flash card in slot 1 on a Management IV module to the SCP-enabled client:

```
C:\> scp terry@192.168.1.50:b:/config2.cfg c:\cfg\config2.cfg
```

Chapter 4
Configuring Secure Shell for Terathon Devices

NOTE: This chapter applies to the BigIron MG8 and NetIron 40G running software release 02.1.00 and later.

Secure Shell (SSH) is a mechanism for allowing secure remote access to management functions on an Foundry device. SSH provides a function similar to Telnet. Users can log into and configure the device using a publicly or commercially available SSH client program, just as they can with Telnet. However, unlike Telnet, which provides no security, SSH provides a secure, encrypted connection to the device.

SSH v2 supported on the BigIron MG8 and NetIron 40G Foundry is compatible with all versions of the SSHv2 protocol (2.1, 2.2, and so on). At the beginning of an SSH session, the Foundry device negotiates the version of SSHv2 to be used. The highest version of SSHv2 supported by both the Foundry device and the client is the version that is used for the session. Once the SSHv2 version is negotiated, the encryption algorithm with the highest security ranking is selected to be used for the session.

Also, BigIron MG8 and NetIron 40G support Secure Copy (SCP) for securely transferring files between a Foundry device and an SCP-enabled remote hosts. See “Using Secure Copy” on page 4-8 for more information.

SSH Version 2 Support

SSHv2 is a substantial revision of Secure Shell, comprising the following hybrid protocols and definitions:

- SSH Transport Layer Protocol
- SSH Authentication Protocol
- SSH Connection Protocol
- SECSH Public Key File Format
- SSH Fingerprint Format
- SSH Protocol Assigned Numbers
- SSH Transport Layer Encryption Modes
- SCP/SFTP/SSH URI Format

If you are using redundant management modules, you can synchronize the DSA host key pair between the active and standby modules by entering the **sync-standby** command at the Privileged EXEC level of the CLI.

Tested SSHv2 Clients

The following SSH clients have been tested with SSHv2:

- SSH Secure Shell 3.2.3

- Van Dyke SecureCRT 4.0 and 4.1
- F-Secure SSH Client 5.3 and 6.0
- PuTTY 0.54 and 0.56
- OpenSSH 3.5_p1 and 3.6.1p2
- Solaris Sun-SSH-1.0

Supported Encryption Algorithms for SSHv2

3DES is the encryption algorithms supported in Foundry implementation of SSHv2.

Supported MAC (Message Authentication Code) Algorithms

SHA 1 is the MAC algorithm supported in Foundry implementation of SSHv2:

Configuring SSH

Foundry's implementation of SSH supports two kinds of user authentication:

- ***DSA challenge-response authentication***, where a collection of public keys are stored on the device. Only clients with a private key that corresponds to one of the stored public keys can gain access to the device using SSH.
- ***Password authentication***, where users attempting to gain access to the device using an SSH client are authenticated with passwords stored on the device or on a TACACS/TACACS+ or RADIUS server

Both kinds of user authentication are enabled by default. You can configure the device to use one or both of them.

To configure Secure Shell on an Foundry device, do the following:

1. Generate a host DSA public and private key pair for the device.
2. Configure DSA challenge-response authentication.
3. Set optional parameters.

You can also view information about active SSH connections on the device as well as terminate them.

Generating a Host Key Pair

When SSH is configured, a public and private ***host DSA key pair*** is generated for the Foundry device. The SSH server on the Foundry device uses this host DSA key pair, along with a dynamically generated ***server DSA key pair***, to negotiate a session key and encryption method with the client trying to connect to it.

The host DSA key pair is stored in the Foundry device's system-config file. Only the public key is readable. The public key should be added to a "known hosts" file (for example, $HOME/.ssh/known_hosts on UNIX systems) on the clients who want to access the device. Some SSH client programs add the public key to the known hosts file automatically; in other cases, you must manually create a known hosts file and place the Foundry device's public key in it. See "Providing the Public Key to Clients" on page 4-3 for an example of what to place in the known hosts file.

While the SSH listener exists at all times, sessions can't be started from clients until a key is generated. Once a key is generated, clients can start sessions. The keys are also not displayed in the configuration file by default. To display the keys, use the **ssh show-host-keys** command in Privileged EXEC mode. To generate a public and private DSA host key pair on a BigIron MG8 and NetIron 40G, enter the following commands:

```
BigIron(config)# crypto key generate
```

When a host key pair is generated, it is saved to the flash memory of all management modules.

To disable SSH in SSHv2 on a BigIron MG8 and NetIron 40G, enter the following commands:

```
BigIron(config)# crypto key zeroize
```

When SSH is disabled, it is deleted from the flash memory of all management modules.

Syntax: crypto key generate | zeroize

The **generate** keyword places an DSA host key pair in the flash memory and enables SSH on the device.

The **zeroize** keyword deletes the DSA host key pair from the flash memory and disables SSH on the device.

By default, public keys are hidden in the running configuration. You can optionally configure the Foundry device to display the DSA host key pair in the running configuration file entering the following command:

```
BigIron# ssh show-host-keys
```

Syntax: ssh show-host-keys

To hide the public keys in the running configuration file, enter the following command:

```
BigIron# ssh no-show-host-keys
```

Syntax: ssh no-show-host-keys

Providing the Public Key to Clients

If you are using SSH to connect to a Foundry device from a UNIX system, you may need to add the Foundry device's public key to a "known hosts" file; for example, $HOME/.ssh/known_hosts. The following is an example of an entry in a known hosts file:

```
AAAAB3NzaC1kc3MAAACBAPY8ZOHY2yFSJA6XYC9HRwNHxaehvx5wOJ0rzZdzoSOXxbET W6ToHv8D1UJ/
z+zHo9Fiko5XybZnDIaBDHtblQ+Yp7StxyltHnXF1YLfKD1G4T6JYrdH YI14Om
1eg9e4NnCRleaqoZPF3UGfZia6bXrGTQf3gJq2e7Yisk/gF+1VAAAAFQDb8D5cv
wHWTZDPfX0D2s9Rd7NBvQAAAIEAlN92+Bb7D4KLYk3IwRbXblwXdkPggA4pfdtW9v
GfJ0/RHd+NjB4eo1D+0dix6tXwYGN7PKS5R/FXPNwxHPapcj9uL1Jn2AWQ2dsknf+i/FAA
vioUPkmdMc0zuWoSOEsSNhVDtX3WdvVcGcBq9cetzrtOKWOocJmJ80qadxTRHtUAAACB
AN7CY+KKv1gHpRzFwdQm7HK9bb1LAo2KwaoXnadFgeptNBQeSXG1vO+JsvphVMBJc9HS
n24VYtYtsMu74qXviYjziVucWKjjKEb11juqnF0GDlB3VVmxHLmxnAz643WK42Z7dLM5
sY29ouezv4Xz2PuMch5VGPP+CDqzCM4loWgV
```

Configuring DSA Challenge-Response Authentication

With DSA challenge-response authentication, a collection of clients' public keys are stored on the Foundry device. Clients are authenticated using these stored public keys. Only clients that have a private key that corresponds to one of the stored public keys can gain access to the device using SSH.

When DSA challenge-response authentication is enabled, the following events occur when a client attempts to gain access to the device using SSH:

1. The client sends its public key to the Foundry device.
2. The Foundry device compares the client's public key to those stored in memory.
3. If there is a match, the Foundry device uses the public key to encrypt a random sequence of bytes.
4. The Foundry device sends these encrypted bytes to the client.
5. The client uses its private key to decrypt the bytes.
6. The client sends the decrypted bytes back to the Foundry device.
7. The Foundry device compares the decrypted bytes to the original bytes it sent to the client. If the two sets of bytes match, it means that the client's private key corresponds to an authorized public key, and the client is authenticated.

Setting up DSA challenge-response authentication consists of the following steps:

8. Importing authorized public keys into the BigIron MG8 and NetIron 40G.
9. Enabling DSA challenge response authentication

Importing Authorized Public Keys into the Foundry Device

SSH clients that support DSA authentication normally provide a utility to generate an DSA key pair. The private key is usually stored in a password-protected file on the local host; the public key is stored in another file and is not protected. You should collect one public key from each client to be granted access to the Foundry device and place all of these keys into one file. This public key file is imported into the Foundry device.

The following is an example of a public key file containing one public keys:

```
---- BEGIN SSH2 PUBLIC KEY ----
Comment: DSA Public Key
AAAAB3NzaC1kc3MAAACBAPY8ZOHY2yFSJA6XYC9HRwNHxaehvx5wOJ0rzZdzoSOXxbET W6ToHv8D1UJ/
z+zHo9Fiko5XybZnDIaBDHtblQ+Yp7StxyltHnXF1YLfKD1G4T6JYrdH YI14Om
1eg9e4NnCRleaqoZPF3UGfZia6bXrGTQf3gJq2e7Yisk/gF+1VAAAAFQDb8D5cv
wHWTZDPfX0D2s9Rd7NBvQAAAIEAlN92+Bb7D4KLYk3IwRbXblwXdkPggA4pfdtW9v
GfJ0/RHd+NjB4eo1D+0dix6tXwYGN7PKS5R/FXPNwxHPapcj9uL1Jn2AWQ2dsknf+i/FAA
vioUPkmdMc0zuWoSOEsSNhVDtX3WdvVcGcBq9cetzrtOKWOocJmJ80qadxTRHtUAAACB
AN7CY+KKv1gHpRzFwdQm7HK9bb1LAo2KwaoXnadFgeptNBQeSXG1vO+JsvphVMBJc9HS
n24VYtYtsMu74qXviYjziVucWKjjKEb11juqnF0GDlB3VVmxHLmxnAz643WK42Z7dLM5
sY29ouezv4Xz2PuMch5VGPP+CDqzCM4loWgV
---- END SSH2 PUBLIC KEY ----
```

You can import the authorized public keys into the active configuration by loading them from a file on a TFTP server and are saved on the EEPROM of the chassis. If you import a public key file from a TFTP server, the file is automatically loaded into the active configuration the next time the device is booted.

To cause a public key file called pkeys.txt to be loaded from a TFTP server each time the Foundry device is booted, enter a command such as the following:

```
BigIron(config)# ip ssh pub-key-file tftp 192.168.1.234 pkeys.txt
```

Syntax: ip ssh pub-key-file tftp ipv6 <ipv6-addr> | <tftp-server-ip-addr> <filename> [remove]

The <tftp-server-ip-addr> variable is the IP address of the tftp server that contains the public key file that you want to import into the Foundry device.

The <filename> variable is the name of the dsa public key file that you want to import into the Foundry device.

The **remove** parameter deletes the key from the system.

To display the currently loaded public keys, enter the following command:

```
BigIron# show ip client-pub-key

---- BEGIN SSH2 PUBLIC KEY ----
Comment: DSA Public Key
AAAAB3NzaC1kc3MAAACBAPY8ZOHY2yFSJA6XYC9HRwNHxaehvx5wOJ0rzZdzoSOXxbET W6ToHv8D1UJ/
z+zHo9Fiko5XybZnDIaBDHtblQ+Yp7StxyltHnXF1YLfKD1G4T6JYrdH YI14Om
1eg9e4NnCRleaqoZPF3UGfZia6bXrGTQf3gJq2e7Yisk/gF+1VAAAAFQDb8D5cv
wHWTZDPfX0D2s9Rd7NBvQAAAIEAlN92+Bb7D4KLYk3IwRbXblwXdkPggA4pfdtW9v
GfJ0/RHd+NjB4eo1D+0dix6tXwYGN7PKS5R/FXPNwxHPapcj9uL1Jn2AWQ2dsknf+i/FAA
vioUPkmdMc0zuWoSOEsSNhVDtX3WdvVcGcBq9cetzrtOKWOocJmJ80qadxTRHtUAAACB
AN7CY+KKv1gHpRzFwdQm7HK9bb1LAo2KwaoXnadFgeptNBQeSXG1vO+JsvphVMBJc9HS
n24VYtYtsMu74qXviYjziVucWKjjKEb11juqnF0GDlB3VVmxHLmxnAz643WK42Z7dLM5
sY29ouezv4Xz2PuMch5VGPP+CDqzCM4loWgV
---- END SSH2 PUBLIC KEY ----
```

Syntax: show ip client-pub-key [| begin<expression> | exclude <expression> | include <expression>]

To clear the public keys from the buffers, enter the following command:

```
BigIron# clear public-key
```

Syntax: clear public-key

Use the **ip ssh pub-key remove** command to delete the public key from the system.

Enabling DSA Challenge-Response Authentication

DSA challenge-response authentication is enabled by default. You can disable or re-enable it manually.

To enable DSA challenge-response authentication:

```
BigIron(config)# ip ssh key-authentication yes
```

To disable DSA challenge-response authentication:

```
BigIron(config)# ip ssh key-authentication no
```

Syntax: ip ssh key-authentication yes | no

Setting the Number of SSH Authentication Retries

By default, the Foundry device attempts to negotiate a connection with the connecting host three times. The number of authentication retries can be changed to between 1 – 5.

For example, the following command changes the number of authentication retries to 5:

```
BigIron(config)# ip ssh authentication-retries 5
```

Syntax: ip ssh authentication-retries <number>

Deactivating User Authentication

After the SSH server on the Foundry device negotiates a session key and encryption method with the connecting client, user authentication takes place. Foundry's implementation of SSH supports DSA challenge-response authentication and password authentication.

With DSA challenge-response authentication, a collection of clients' public keys are stored on the Foundry device. Clients are authenticated using these stored public keys. Only clients that have a private key that corresponds to one of the stored public keys can gain access to the device using SSH.

With password authentication, users are prompted for a password when they attempt to log into the device (provided empty password logins are not allowed; see "Enabling Empty Password Logins" on page 4-5). If there is no user account that matches the user name and password supplied by the user, the user is not granted access.

You can deactivate one or both user authentication methods for SSH. Note that deactivating both authentication methods essentially disables the SSH server entirely.

To disable DSA challenge-response authentication:

```
BigIron(config)# ip ssh key-authentication no
```

Syntax: ip ssh key-authentication no | yes

The default is "yes".

To deactivate password authentication:

```
BigIron(config)# ip ssh password-authentication no
```

Syntax: ip ssh password-authentication no | yes

The default is "yes".

Enabling Empty Password Logins

By default, empty password logins are not allowed. This means that users with an SSH client are always prompted for a password when they log into the device. To gain access to the device, each user must have a user name and password. Without a user name and password, a user is not granted access. See "Setting Up Local User Accounts" on page 2-19 for information on setting up user names and passwords.

If you enable empty password logins, users are ***not*** prompted for a password when they log in. Any user with an SSH client can log in without being prompted for a password.

To enable empty password logins:

```
BigIron(config)# ip ssh permit-empty-passwd yes
```

Syntax: ip ssh permit-empty-passwd no | yes

Setting the SSH Port Number

By default, SSH traffic occurs on TCP port 22. You can change this port number. For example, the following command changes the SSH port number to 2200:

```
BigIron(config)# ip ssh port 2200
```

Note that if you change the default SSH port number, you must configure SSH clients to connect to the new port. Also, you should be careful not to assign SSH to a port that is used by another service. If you change the SSH port number, Foundry recommends that you change it to a port number greater than 1024.

Syntax: ip ssh port <number>

Setting the SSH Login Timeout Value

When the SSH server attempts to negotiate a session key and encryption method with a connecting client, it waits a maximum of 120 seconds for a response from the client. If there is no response from the client after 120 seconds, the SSH server disconnects. You can change this timeout value to between 1 – 120 seconds. For example, to change the timeout value to 60 seconds:

```
BigIron(config)# ip ssh timeout 60
```

Syntax: ip ssh timeout <seconds>

Designating an Interface as the Source for All SSH Packets

You can designate a loopback interface, virtual interface, POS port, or Ethernet port as the source for all SSH packets from the device. The software uses the IP address with the numerically lowest value configured on the port or interface as the source IP address for SSH packets originated by the device.

NOTE: When you specify a single SSH source, you can use only that source address to establish SSH management sessions with the Foundry device.

To specify the numerically lowest IP address configured on a loopback interface as the device's source for all SSH packets, enter commands such as a the following:

```
BigIron(config)# int loopback 2
BigIron(config-lbif-2)# ip address 10.0.0.2/24
BigIron(config-lbif-2)# exit
BigIron(config)# ip ssh source-interface loopback 2
```

The commands in this example configure loopback interface 2, assign IP address 10.0.0.2/24 to the interface, then designate the interface as the source for all SSH packets from the BigIron MG8 and NetIron 40G.

Syntax: ip ssh source-interface ethernet <slot/port> | loopback <num> | ve <num>

The <num> parameter is a loopback interface or virtual interface number. The <slot/port> parameter specifies an ethernet port number. For example:

```
BigIron(config)# interface ethernet 1/4
BigIron(config-if-1/4)# ip address 209.157.22.110/24
BigIron(config-if-1/4)# exit
BigIron(config)# ip ssh source-interface ethernet 1/4
```

Configuring Maximum Idle Time for SSH Sessions

By default, SSH sessions do not time out. Optionally, you can set the amount of time an SSH session can be inactive before the Foundry device closes it. For example, to set the maximum idle time for SSH sessions to 30 minutes:

```
BigIron(config)# ip ssh idle-time 30
```

Syntax: ip ssh idle-time <minutes>

If an established SSH session has no activity for the specified number of minutes, the Foundry device closes it. An idle time of 0 minutes (the default value) means that SSH sessions never time out. The maximum idle time for SSH sessions is 240 minutes.

Filtering SSH Access Using ACLs

You can permit or deny SSH access to the Foundry device using ACLs. To use ACLs, first create the ACLs you want to use. You can specify a numbered standard IPv4 ACL, a named standard IPv4 ACL, or an IPv6 ACL.

Then enter the following command:

```
BigIron(config)# access-list 10 permit host 192.168.144.241
BigIron(config)# access-list 10 deny host 192.168.144.242 log
BigIron(config)# access-list 10 permit host 192.168.144.243
BigIron(config)# access-list 10 deny any
BigIron(config)# ssh access-group 10
```

Syntax: ssh access-group <standard-named-acl> | <standard-numbered-acl> | <ipv6-acl>

Displaying SSH Connection Information

Up to five SSH connections can be active on the Foundry device. To display information about SSH connections, enter the following command:

```
BigIron# show ip ssh
Connection Version  Encryption  Username
1           SSH-2   3des-cbc    Hanuma
2           SSH-2   3des-cbc    Mikaila
3           SSH-2   3des-cbc    Jenny
4           SSH-2   3des-cbc    Mariah
5           SSH-2   3des-cbc    Logan
```

Syntax: show ip ssh [| begin <expression> | exclude <expression> | include <expression>]

This display shows the following information about the active SSH connections:

Table 4.1: SSH Connection Information

This Field...	Displays...
Connection	The SSH connection ID. This can be from 1 – 5.
Version	The SSH version number. This should always be 1.5.
Encryption	The encryption method used for the connection.
Username	The user name for the connection.

The **show who** command also displays information about SSH connections. For example:

```
BigIron#show who
Console connections:
established, monitor enabled, in config mode
2 minutes 17 seconds in idle
Telnet connections (inbound):
1 closed
2 closed
3 closed
4 closed
5 closed
Telnet connection (outbound):
6 closed
SSH connections:
1 established, client ip address 192.168.144.241, user is hanuma
1 minutes 16 seconds in idle
2 established, client ip address 192.168.144.241, user is Mikaila
you are connecting to this session
18 seconds in idle
3 established, client ip address 192.168.144.241, user is Jenny
1 minutes 39 seconds in idle
4 established, client ip address 192.168.144.242, user is Mariah
41 seconds in idle
5 established, client ip address 192.168.144.241, user is Logan
23 seconds in idle
```

Syntax: show who [| begin<expression> | exclude<expression> | include<expression>]

To terminate one of the active SSH connections, enter the following command:

```
BigIron# kill ssh 1
```

Syntax: kill ssh <connection-id>

Using Secure Copy

Secure Copy (SCP) uses security built into SSH to transfer files between hosts on a network, providing a more secure file transfer method than Remote Copy (RCP) or FTP. SCP automatically uses the authentication methods, encryption algorithm, and data compression level configured for SSH. For example, if password authentication is enabled for SSH, the user is prompted for a user name and password before SCP allows a file to be transferred. No additional configuration is required for SCP on top of SSH.

You can use SCP to copy files on the Foundry device, including the startup configuration and running configuration files, to or from an SCP-enabled remote host.

SCP is enabled by default and can be disabled. To disable SCP, enter the following command:

```
BigIron(config)# ip ssh scp disable
```

Syntax: ip ssh scp disable | enable

NOTE: If you disable SSH, SCP is also disabled.

The following are examples of using SCP to transfer files from and to an Foundry device

NOTE: When using SCP, you enter the **scp** commands on the SCP-enabled client, rather than the console on the Foundry device.

NOTE: Certain SCP client options, including -p and -r, are ignored by the SCP server on the Foundry device. If an option is ignored, the client is notified.

To copy a configuration file (c:\cfg\foundryhp.cfg) to the running configuration file on an Foundry device at 192.168.1.50 and log in as user terry, enter the following command on the SCP-enabled client:

```
C:\> scp c:\cfg\foundryhp.cfg terry@192.168.1.50:runConfig
```

If password authentication is enabled for SSH, the user is prompted for user terry's password before the file transfer takes place.

To copy the configuration file to the startup configuration file:

```
C:\> scp c:\cfg\foundryhp.cfg terry@192.168.1.50:startConfig
```

To copy the configuration file to a file called config1.cfg on the PCMCIA flash card in slot 1 on a management module:

```
C:\> scp c:\cfg\foundry.cfg terry@192.168.1.50:slot1:/config1.cfg
```

To copy the configuration file to a file called config1.cfg on the PCMCIA flash card in slot 2 on a management module:

```
C:\> scp c:\cfg\foundry.cfg terry@192.168.1.50:slot2:/config1.cfg
```

To copy the running configuration file on an Foundry device to a file called c:\cfg\fdryhprun.cfg on the SCP-enabled client:

```
C:\> scp terry@192.168.1.50:runConfig c:\cfg\fdryhprun.cfg
```

To copy the startup configuration file on an Foundry device to a file called c:\cfg\fdryhpstart.cfg on the SCP-enabled client:

```
C:\> scp terry@192.168.1.50:startConfig c:\cfg\fdryhpstart.cfg
```

Chapter 5
Configuring 802.1X Port Security

Overview

NOTE: This feature is available on devices running Enterprise software release 07.6.01 and later, and on the FES and all and FastIron X-Series devices. On BigIron MG8 and NetIron 40G, this feature is introduced in software release 02.1.00 and later.

Foundry devices support the IEEE 802.1X standard for authenticating devices attached to LAN ports. Using 802.1X port security, you can configure a Foundry device to grant access to a port based on information supplied by a client to an authentication server.

When a user logs on to a network that uses 802.1X port security, the Foundry device grants (or doesn't grant) access to network services after the user is authenticated by an authentication server. The user-based authentication in 802.1X port security provides an alternative to granting network access based on a user's IP address, MAC address, or subnetwork.

This chapter is divided into the following sections:

- "How 802.1X Port Security Works" on page 5-1 explains basic concepts about 802.1X port security.
- "Configuring 802.1X Port Security" on page 5-10 describes how to set up 802.1X port security on Foundry devices using the Command Line Interface (CLI).
- "Displaying 802.1X Information" on page 5-29 describes the commands used to display information about an 802.1X port security configuration.
- "Sample 802.1X Configurations" on page 5-41 shows diagrams of two 802.1X port security configurations and the CLI commands used for implementing them.

IETF RFC Support

Foundry's implementation of 802.1X port security supports the following RFCs:

- RFC 2284 PPP Extensible Authentication Protocol (EAP)
- RFC 2865 Remote Authentication Dial In User Service (RADIUS)
- RFC 2869 RADIUS Extensions

How 802.1X Port Security Works

This section explains the basic concepts behind 802.1X port security, including device roles, how the devices communicate, and the procedure used for authenticating clients.

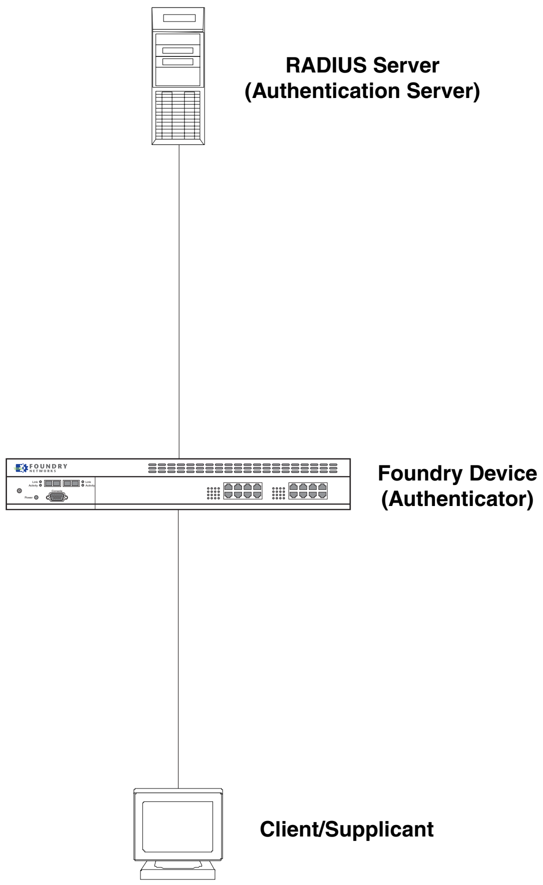
Device Roles in an 802.1X Configuration

The 802.1X standard defines the roles of ***Client/Supplicant***, ***Authenticator***, and ***Authentication Server*** in a network.

The Client (known as a ***Supplicant*** in the 802.1X standard) provides username/password information to the Authenticator. The Authenticator sends this information to the Authentication Server. Based on the Client's information, the Authentication Server determines whether the Client can use services provided by the Authenticator. The Authentication Server passes this information to the Authenticator, which then provides services to the Client, based on the authentication result.

Figure 5.1 illustrates these roles.

Figure 5.1 Authenticator, Client/Supplicant, and Authentication Server in an 802.1X configuration



Authenticator – The device that controls access to the network. In an 802.1X configuration, the Foundry device serves as the Authenticator. The Authenticator passes messages between the Client and the Authentication Server. Based on the identity information supplied by the Client, and the authentication information supplied by the Authentication Server, the Authenticator either grants or does not grant network access to the Client.

Client/Supplicant – The device that seeks to gain access to the network. Clients must be running software that supports the 802.1X standard (for example, the Windows XP operating system). Clients can either be directly connected to a port on the Authenticator, or can be connected by way of a hub.

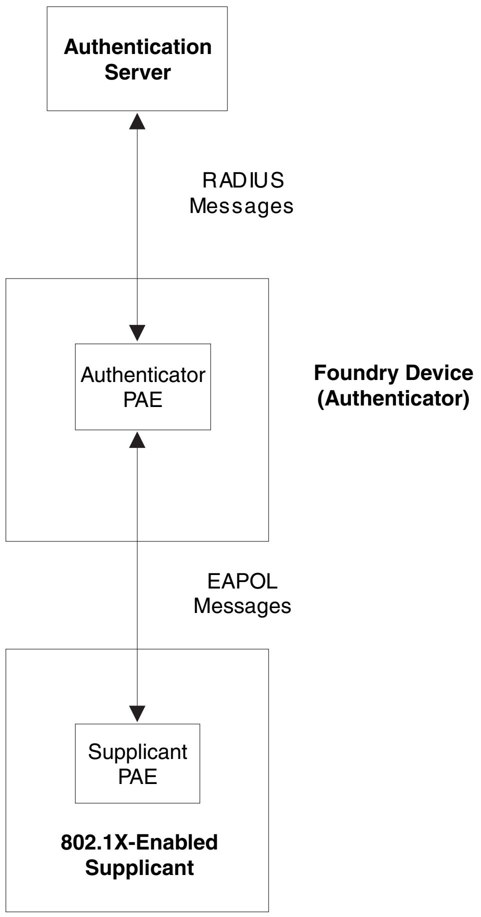
Authentication Server – The device that validates the Client and specifies whether or not the Client may access services on the device. Foundry supports Authentication Servers running RADIUS.

Communication Between the Devices

For communication between the devices, 802.1X port security uses the ***Extensible Authentication Protocol*** (EAP), defined in RFC 2284. The 802.1X standard specifies a method for encapsulating EAP messages so that they can be carried over a LAN. This encapsulated form of EAP is known as EAP over LAN (***EAPOL***). The standard also specifies a means of transferring the EAPOL information between the Client/Supplicant, Authenticator, and Authentication Server.

EAPOL messages are passed between the ***Port Access Entity (PAE)*** on the Supplicant and the Authenticator. Figure 5.2 shows the relationship between the Authenticator PAE and the Supplicant PAE.

Figure 5.2 Authenticator PAE and Supplicant PAE



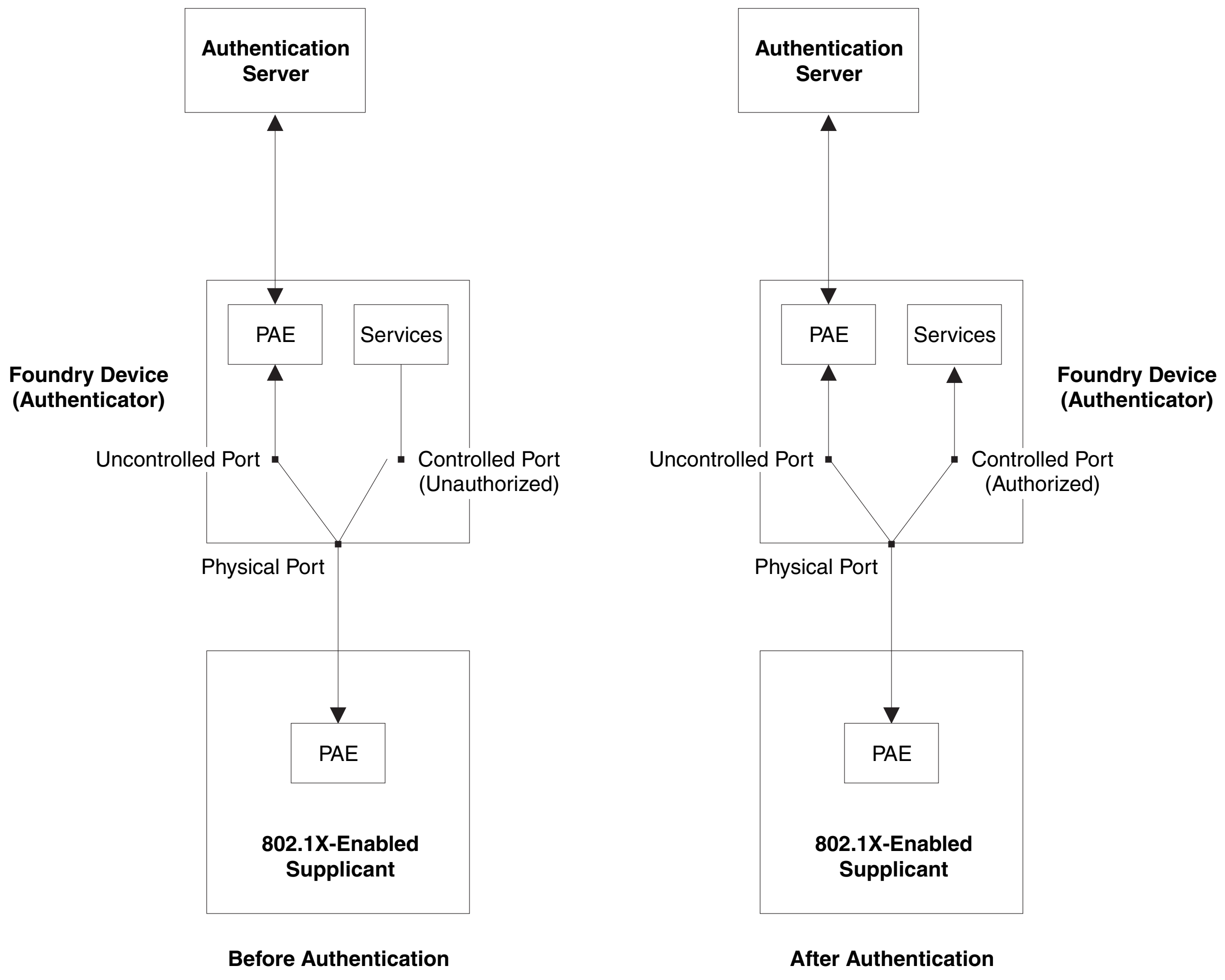
Authenticator PAE – The Authenticator PAE communicates with the Supplicant PAE, receiving identifying information from the Supplicant. Acting as a RADIUS client, the Authenticator PAE passes the Supplicant's information to the Authentication Server, which decides whether the Supplicant can gain access to the port. If the Supplicant passes authentication, the Authenticator PAE grants it access to the port.

Supplicant PAE – The Supplicant PAE supplies information about the Client to the Authenticator PAE and responds to requests from the Authenticator PAE. The Supplicant PAE can also initiate the authentication procedure with the Authenticator PAE, as well as send logoff messages.

Controlled and Uncontrolled Ports

A physical port on the device used with 802.1X port security has two virtual access points: a ***controlled*** port and an ***uncontrolled*** port. The controlled port provides full access to the network. The uncontrolled port provides access only for EAPOL traffic between the Client and the Authentication Server. When a Client is successfully authenticated, the controlled port is opened to the Client. Figure 5.3 illustrates this concept.

Figure 5.3 Controlled and Uncontrolled Ports before and after Client authentication



Before a Client is authenticated, only the uncontrolled port on the Authenticator is open. The uncontrolled port allows only EAPOL frames to be exchanged between the Client and the Authentication Server. The controlled port is in the unauthorized state and allows no traffic to pass through.

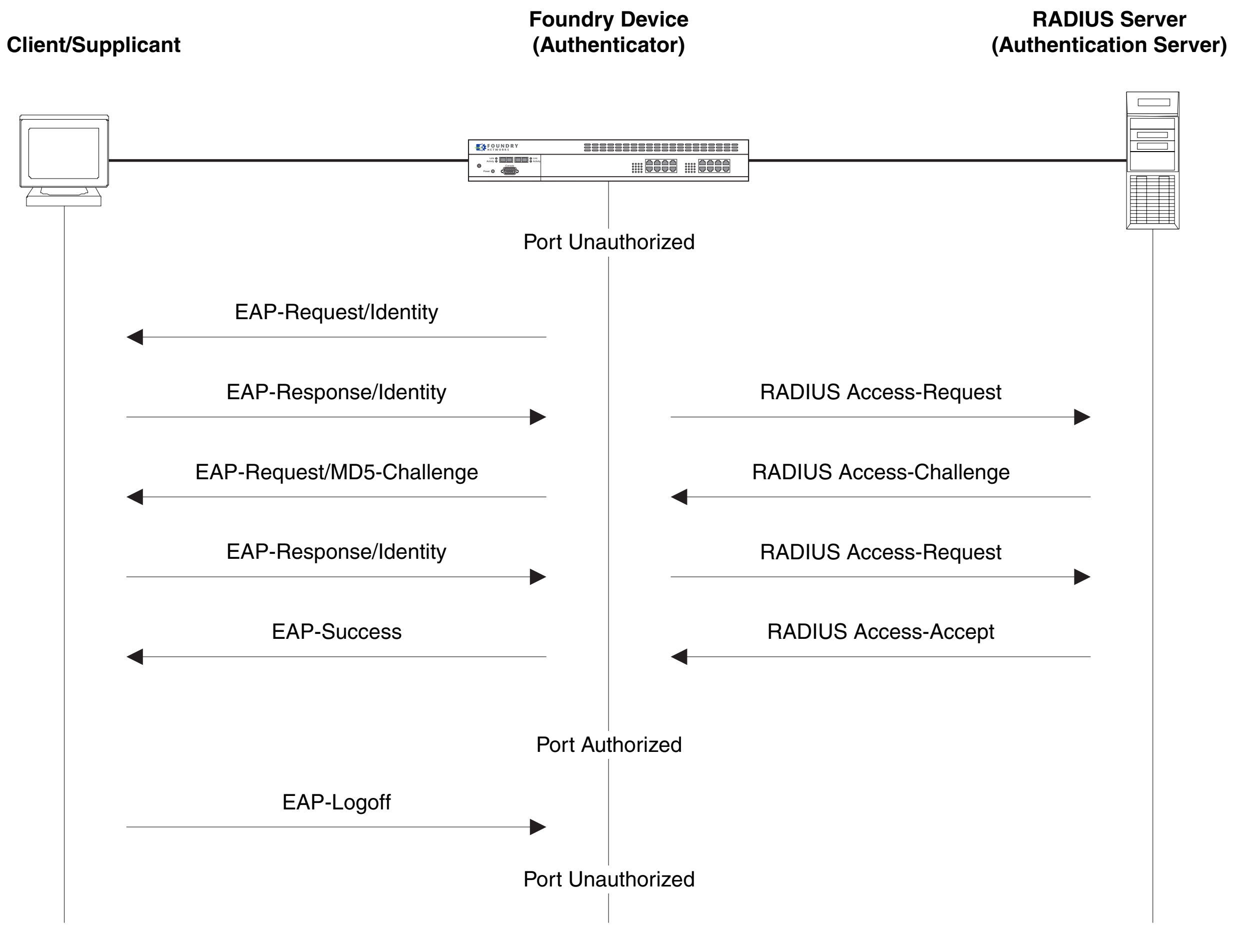
During authentication, EAPOL messages are exchanged between the Supplicant PAE and the Authenticator PAE, and RADIUS messages are exchanged between the Authenticator PAE and the Authentication Server. See "Message Exchange During Authentication" on page 5-4 for an example of this process. If the Client is successfully authenticated, the controlled port becomes authorized, and traffic from the Client can flow through the port normally.

By default, all controlled ports on the Foundry device are placed in the authorized state, allowing all traffic. When authentication is activated on an 802.1X-enabled interface, the interface's controlled port is placed initially in the unauthorized state. When a Client connected to the port is successfully authenticated, the controlled port is then placed in the authorized state until the Client logs off. See "Enabling 802.1X Port Security" on page 5-21 for more information.

Message Exchange During Authentication

Figure 5.4 illustrates a sample exchange of messages between an 802.1X-enabled Client, a Foundry device acting as Authenticator, and a RADIUS server acting as an Authentication Server.

Figure 5.4 Message exchange between Client/Supplicant, Authenticator, and Authentication Server



In this example, the Authenticator (the Foundry device) initiates communication with an 802.1X-enabled Client. When the Client responds, it is prompted for a username (255 characters maximum) and password. The Authenticator passes this information to the Authentication Server, which determines whether the Client can access services provided by the Authenticator. When the Client is successfully authenticated by the RADIUS server, the port is authorized. When the Client logs off, the port becomes unauthorized again.

Starting in release 07.6.03, Foundry's 802.1X implementation supports dynamic VLAN assignment. If one of the attributes in the Access-Accept message sent by the RADIUS server specifies a VLAN identifier, and this VLAN is available on the Foundry device, the client's port is moved from its default VLAN to the specified VLAN. When the client disconnects from the network, the port is placed back in its default VLAN. See "Configuring Dynamic VLAN Assignment for 802.1X Ports" on page 5-12 for more information.

If a Client does not support 802.1X, authentication cannot take place. The Foundry device sends EAP-Request/ Identity frames to the Client, but the Client does not respond to them.

When a Client that supports 802.1X attempts to gain access through a non-802.1X-enabled port, it sends an EAP start frame to the Foundry device. When the device does not respond, the Client considers the port to be authorized, and starts sending normal traffic.

Foundry devices support Identity and MD5-challenge request types in EAP Request/Response messages. However, FES devices running software release 03.2.00, FESX devices running software release 02.1.01, FSX devices running software release 02.3.01, and devices running Enterprise software release 07.8.00 support the following 802.1X authentication challenge types:

NOTE: See also "EAP Pass-Through Support" on page 5-6.

- EAP-TLS (RFC 2716) – EAP Transport Level Security (TLS) provides strong security by requiring both client and authentication server to be identified and validated through the use of public key infrastructure (PKI) digital certificates. EAP-TLS establishes a tunnel between the client and the authentication server to protect messages from unauthorized users' eavesdropping activities. Since EAP-TLS requires PKI digital certificates on both the clients and the authentication servers, the roll out, maintenance, and scalability of this authentication method is much more complex than other methods. EAP-TLS is best for installations with existing PKI certificate infrastructures.
- EAP-TTLS (Internet-Draft) – The EAP Tunneled Transport Level Security (TTLS) is an extension of EAP-TLS Like TLS, EAP-TTLS provides strong authentication; however it requires only the authentication server to be validated by the client through a certificate exchange between the server and the client. Clients are authenticated by the authentication server using user names and passwords.

  A TLS tunnel can be used to protect EAP messages and existing user credential services such as Active Directory, RADIUS, and LDAP. Backward compatibility for other authentication protocols such as PAP, CHAP, MS-CHAP, and MS-CHAP-V2 are also provided by EAP-TTLS. EAP-TTLS is not considered foolproof and can be fooled into sending identity credentials if TLS tunnels are not used. EAP-TTLS is suited for installations that require strong authentication without the use of mutual PKI digital certificates.
- PEAP (Internet-Draft) – Protected EAP Protocol (PEAP) is an Internet-Draft that is similar to EAP-TTLS. PEAP client authenticates directly with the backend authentication server. The authenticator acts as a pass-through device, which does not need to understand the specific EAP authentication protocols.

  Unlike EAP-TTLS, PEAP does not natively support user name and password to authenticate clients against an existing user database such as LDAP. PEAP secures the transmission between the client and authentication server with a TLS encrypted tunnel. PEAP also allows other EAP authentication protocols to be used. It relies on the mature TLS keying method for its key creation and exchange. PEAP is best suited for installations that require strong authentication without the use of mutual certificates.

NOTE: If the 802.1X Client will be sending a packet that is larger than 1500 bytes, then the following must be configured on the Foundry device:

- On devices with JetCore modules, **default-mtu 1700** must be configured.
- On devices with IronCore modules, **jumbo 1920** must be configured.
- On FES devices, enable **jumbo2048** at the Global CONFIG level of the CLI.

Configuration for these challenge types is the same as for the EAP-MD5 challenge type.

EAP Pass-Through Support

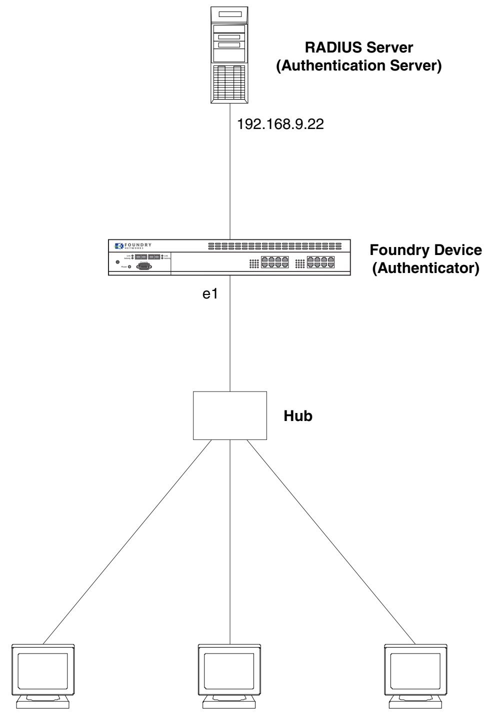
Starting with FES software release 03.4.00, and FESX/FSX/FWSX software release 02.3.01, EAP pass-through support is fully compliant with RFC 3748, in which, by default, compliant pass-through authenticator implementations forward EAP challenge request packets of any type. Previous software releases were limited to MD5, EAP-TTLS, EAP-PEAP, and EAP-TLS challenge request types for 802.1X authentication.

NOTE: If the 802.1X supplicant or authentication server will be sending packets that are greater than 1500 MTU, you should configure the FES device to accommodate a bigger buffer size.

Authenticating Multiple Hosts Connected to the Same Port

Foundry devices support 802.1X authentication for ports with more than one host connected to them. Figure 5.5 illustrates a sample configuration where multiple hosts are connected to a single 802.1X port.

Figure 5.5 Multiple hosts connected to a single 802.1X-enabled port



Clients/Supplicants running 802.1X-compliant client software

The way the Foundry device authenticates Clients in a multiple-host configuration depends on the software release running on the device:

- In releases prior to 07.8.00, services are provided on a port based on the authentication of a single Client. When one Client is successfully authenticated, all hosts connected to the port are allowed access to the network. The Foundry device forwards traffic from all of the connected hosts for as long as the authenticated Client stays connected. When the authenticated Client disconnects from the network, authentication is removed for the other connected hosts as well.
- Starting in release 03.3.00 for the FES, release 02.2.00 for the FESX and FWSX, release 02.3.01 for the FSX, and Enterprise release 07.8.00, if there are multiple hosts connected to a single 802.1X-enabled port, the Foundry device authenticates each of them individually. Each host's authentication status is independent of the others, so that if one authenticated host disconnects from the network, it has no effect on the authentication status of any of the other authenticated hosts.

  By default, traffic from hosts that cannot be authenticated by the RADIUS server is dropped in hardware. You can optionally configure the Foundry device to assign the port to a "restricted" VLAN if authentication of the Client is unsuccessful.

How 802.1X Multiple-Host Authentication Works

NOTE: This section applies to release 03.3.00 and later for the FES, release 02.2.00 and later for the FESX and FWSX, release 02.3.01 and later for the FSX, Enterprise release 07.8.00 and later, and release 02.1.00 and later for the BigIron MG8 and NetIron 40G.

When multiple hosts are connected to a single 802.1X-enabled port on a Foundry device (as in Figure 5.5), 802.1X authentication is performed in the following way:

1. One of the 802.1X-enabled Clients attempts to log into a network in which a Foundry device serves as an Authenticator.

2. The Foundry device creates an internal session (called a ***dot1x-mac-session***) for the Client. A dot1x-mac-session serves to associate a Client's MAC address and username with its authentication status.
3. The Foundry device performs 802.1X authentication for the Client. Messages are exchanged between the Foundry device and the Client, and between the device and the Authentication Server (RADIUS server). The result of this process is that the Client is either successfully authenticated or not authenticated, based on the username and password supplied by the client.
4. If the Client is successfully authenticated, the Client's dot1x-mac-session is set to "access-is-allowed". This means that traffic from the Client can be forwarded normally.
5. If authentication for the Client is unsuccessful the first time, multiple attempts to authenticate the client will be made as determined by the **attempts** variable in the **auth-fail-max-attempts** command.
   - See "Specifying the Number of Authentication Attempts the Device Makes Before Dropping Packets" on page 5-27 for information on how to do this.
6. If authentication for the Client is unsuccessful more than the number of times specified by the **attempts** variable in the **auth-fail-max-attempts** command, an ***authentication-failure action*** is taken. The authentication-failure action can be either to drop traffic from the Client, or to place the port in a "restricted" VLAN.
   - If the authentication-failure action is to drop traffic from the Client, then the Client's dot1x-mac-session is set to "access-denied", causing traffic from the Client to be dropped in hardware.
   - If the authentication-failure action is to place the port in a "restricted" VLAN, If the Client's dot1x-mac-session is set to "access-restricted" then the port is moved to the specified restricted VLAN, and traffic from the Client is forwarded normally.
7. When the Client disconnects from the network, the Foundry device deletes the Client's dot1x-mac-session. This does not affect the dot1x-mac-session or authentication status (if any) of the other hosts connected on the port.

Notes

- The Client's dot1x-mac-session establishes a relationship between the username and MAC address used for authentication. If a user attempts to gain access from different Clients (with different MAC addresses), he or she would need to be authenticated from each Client.
- If a Client has been denied access to the network (that is, the Client's dot1x-mac-session is set to "access-denied"), then you can cause the Client to be re-authenticated by manually disconnecting the Client from the network, or by using the **clear dot1x mac-session** command. See "Clearing a dot1x-mac-session for a MAC Address" on page 5-28 for information on this command.
- When a Client has been denied access to the network, its dot1x-mac-session is aged out if no traffic is received from the Client's MAC address over a fixed hardware aging period (70 seconds), plus a configurable software aging period. You can optionally change the software aging period for dot1x-mac-sessions or disable aging altogether. After the denied Client's dot1x-mac-session is aged out, traffic from that Client is no longer blocked, and the Client can be re-authenticated.

  In addition, you can configure disable aging for the dot1x-mac-session of Clients that have been granted either full access to the network, or have been placed in a restricted VLAN. After a Client's dot1x-mac-session ages out, the Client must be re-authenticated. See "Disabling Aging for dot1x-mac-sessions" on page 5-27 for more information.
- On some earlier software releases, dynamic IP ACL and MAC address filter assignment is not supported in an 802.1X multiple-host configuration. If a RADIUS server returns an Access-Accept message that specifies an IP ACL or MAC address filter for the Client, these attributes are ignored.

  In Enterprise software release 07.8.01 and later, dynamic IP ACL and MAC address filter assignment is now supported in an 802.1X multiple-host configuration.

  In release 02.1.00 for the BigIron MG8 and NetIron 40G, dynamic IP ACL and MAC address filter assignment is supported to a limited degree in an 802.1X multiple-host configuration. See "On BigIron MG8 and NetIron 40G Running Release 02.1.00 and Later" under "Dynamically Applying IP ACLs and MAC Filters to 802.1X Ports" on page 5-15.

In release 03.3.00 for the FES and release 02.2.00 for the for the FastIron Edge Switch X-Series and FastIron SuperX, dynamic IP ACL and MAC address filter assignment is supported in an 802.1X multiple-host configuration. See “On the FES Running Release 03.3.00 and Later” and “On the FESX, FSX, and FWSX” under “Dynamically Applying IP ACLs and MAC Filters to 802.1X Ports” on page 5-15.

- In release 03.3.00 for the FES, release 02.2.00 for the FESX and FWSX, and release 02.3.01 for the FSX, 802.1X multiple-host authentication has the following additions:
    - Configurable hardware aging period for denied client dot1x-mac-sessions. See “Configurable Hardware Aging Period for Denied Client dot1x-mac-sessions” on page 5-9.
    - Dynamic ACL and MAC address filter assignment in 802.1X multiple-host configurations. See “On the FES Running Release 03.3.00 and Later” under “Dynamically Applying IP ACLs and MAC Filters to 802.1X Ports” on page 5-15.
    - Dynamic multiple VLAN assignment for 802.1X ports. See “Dynamic Multiple VLAN Assignment for 802.1X Ports on the FES, FESX, FSX, and FWSX” on page 5-13.
    - Enhancements to some **show** commands. See “Enhancement to the show Commands” on page 5-40.
    - Differences in command syntax for saving dynamic VLAN assignments to the startup-config file (for the FESX, FSX, and FWSX).

Configurable Hardware Aging Period for Denied Client dot1x-mac-sessions

NOTE: This feature is introduced in FESX and FWSX release 02.2.00, FSX release 02.3.01, and FES release 03.3.00.

When one of the 802.1X-enabled Clients in a multiple-host configuration attempts to log into a network in which an FES or FastIron X-Series device serves as an Authenticator, the device creates a dot1x-mac-session for the Client.

When a Client has been denied access to the network, its dot1x-mac-session is aged out if no traffic is received from the Client’s MAC address over a period of time. After a denied Client’s dot1x-mac-session ages out, the Client can be re-authenticated. Aging of a denied Client's dot1x-mac-session occurs in two phases, known as hardware aging and software aging.

On BigIron/FastIron devices, the hardware aging period for a denied Client's dot1x-mac-session is fixed at 70 seconds and is non-configurable. (The hardware aging time for an authenticated Client's dot1x-mac-session is the length of time specified with the **mac-age** command.) The software aging period for a denied Client's dot1x-mac-session is configurable through the CLI, with the **mac-session-aging max-age** command. Once the hardware aging period ends, the software aging period begins. When the software aging period ends, the denied Client's dot1x-mac-session ages out, and the Client can be authenticated again.

On FES and FastIron X-Series devices, the hardware aging period for a denied Client's dot1x-mac-session is not fixed at 70 seconds. The hardware aging period for a denied Client's dot1x-mac-session is equal to the length of time specified with the **mac-age** command (for the FES) or the dot1x **timeout quiet-period** command (for the FESX, FSX, and FWSX). By default, the hardware aging time is 60 seconds. As on BigIron/FastIron devices, once the hardware aging period ends, the software aging period begins. When the software aging period ends, the denied Client's dot1x-mac-session ages out, and the Client can be authenticated again.

802.1X Port Security and sFlow

sFlow is a system for observing traffic flow patterns and quantities within and among a set of Layer 2 Switches and Layer 3 Switches. sFlow works by taking periodic samples of network data and exporting this information to a collector.

When you enable sFlow forwarding on an 802.1X-enabled interface, the samples taken from the interface include the user name string at the inbound and/or outbound port, if that information is available.

For more information on sFlow, see the “sFlow“ section in the “Remote Network Monitoring“ chapter of the *Foundry Enterprise Configuration and Management Guide*.

Configuring 802.1X Port Security

Configuring 802.1X port security on a Foundry device consists of the following tasks:

1. Configuring the Foundry device's interaction with the Authentication Server:
   - "Configuring an Authentication Method List for 802.1X" on page 5-10
   - "Setting RADIUS Parameters" on page 5-11
   - "Configuring Dynamic VLAN Assignment for 802.1X Ports" on page 5-12 (optional)
   - "Dynamically Applying IP ACLs and MAC Filters to 802.1X Ports" on page 5-15
2. Configuring the Foundry device's role as the Authenticator:
   - "Enabling 802.1X Port Security" on page 5-21
   - "Initializing 802.1X on a Port" on page 5-26 (optional)
3. Configuring the Foundry device's interaction with Clients:
   - "Configuring Periodic Re-Authentication" on page 5-23 (optional)
   - "Re-Authenticating a Port Manually" on page 5-23 (optional)
   - "Setting the Quiet Period" on page 5-24 (optional)
   - "Setting the Wait Interval for EAP Frame Retransmissions" on page 5-24 (optional)
   - "Setting the Maximum Number of EAP Frame Retransmissions" on page 5-24 (optional)
   - "Specifying the Security Hold Time" on page 5-23 (optional)
   - "Specifying a Timeout for Retransmission of Messages to the Authentication Server" on page 5-25 (optional)
   - "Allowing Access to Multiple Hosts" on page 5-26 (optional)
   - "Defining MAC Filters for EAP Frames" on page 5-28 (optional)

Configuring an Authentication Method List for 802.1X

To use 802.1X port security, you must specify an authentication method to be used to authenticate Clients. Foundry supports RADIUS authentication with 802.1X port security. To use RADIUS authentication with 802.1X port security, you create an authentication method list for 802.1X and specify RADIUS as an authentication method, then configure communication between the Foundry device and RADIUS server.

For example:

```
BigIron(config)# aaa authentication dot1x default radius
```

Syntax: [no] aaa authentication dot1x default <method-list>

For the <method-list>, enter at least one of the following authentication methods:

radius – Use the list of all RADIUS servers that support 802.1X for authentication.

none – Use no authentication. The Client is automatically authenticated without the device using information supplied by the Client.

NOTE: If you specify both **radius** and **none**, make sure **radius** comes before **none** in the method list.

Setting RADIUS Parameters

To use a RADIUS server to authenticate access to a Foundry device, you must identify the server to the Foundry device. For example:

```
BigIron(config)# radius-server host 209.157.22.99 auth-port 1812 acct-port 1813
default key mirabeau dot1x
```

```
BigIron(config)# radius-server host 209.157.22.99 auth-port 1812 acct-port 1813
default key mirabeau dot1x
```

Syntax: radius-server host <ip-addr> | <server-name> [authentication-only | accounting-only | default] [key 0 | 1 <string>] [dot1x]

The **host** <ip-addr> **|** <server-name> parameter is either an IP address or an ASCII text string.

The **dot1x** parameter indicates that this RADIUS server supports the 802.1X standard. A RADIUS server that supports the 802.1X standard can also be used to authenticate non-802.1X authentication requests.

NOTE: To implement 802.1X port security, at least one of the RADIUS servers identified to the Foundry device must support the 802.1X standard.

Supported RADIUS Attributes

Many IEEE 802.1X Authenticators will function as RADIUS clients. Some of the RADIUS attributes may be received as part of IEEE 802.1X authetnication. The IronCore, JetCore, FastIron Edge Switch, and FastIron Edge Switch X-Series devices support the following RADIUS attributes for IEEE 802.1X authetnication:

- Username (1) – RFC 2865
- NAS-IP-Address (4) – RFC 2865
- NAS-Port (5) – RFC 2865
- Service-Type (6) – RFC 2865
- FilterId (11) – RFC 2865
- Framed-MTU (12) – RFC 2865
- State (24) – RFC 2865
- Vendor-Specific (26) – RFC 2865
- Session-Timeout (27) – RFC 2865
- Termination-Action (29) – RFC 2865
- Calling-Station-ID (31) – RFC 2865
- NAS-Port-Type (61) š RFC 2865
- Tunnel-Type (64) – RFC 2868
- Tunnel-Medium-Type (65) – RFC 2868
- EAP Message (79) – RFC 2579
- Message-Authenticator (80) RFC 3579
- Tunnel-Private-Group-Id (81) – RFC 2868
- NAS-Port-id (87) – RFC2869

802.1X Termination-Action for RADIUS Clients

NOTE: This attribute is supported on the FES running releases 03.4.00 and later.

Software release 03.4.00 and 802.1X support the *termination-action* attribute, which specifies the action to be taken by the authenticator (the Foundry device) after a service has completed. A termination-action value of 1 (RADIUS-request value) indicates that re-authentication should occur after the session time has expired. A termination-action value of 0 (the default termination-action value) indicates that the session should terminate.

NOTE: Together with the *session-timeout* attribute, these commands can be used to provide quarantined VLAN remediation support.

802.1X Session Timeout for RADIUS Clients

NOTE: This attribute is supported on the FES running release 03.4.00 or later.

Software release 03.4.00 and 802.1X support the *session timeout* attribute, as follows:

- When specified in a RADIUS access-accept message that specifies the default termination-action or no termination-action, the session timeout attribute specifies the maximum number of seconds the authenticator (the Foundry device) will wait before terminating the session.
- When specified in a RADIUS access-accept message that specifies a termination-action value of 1 (RADIUS-request), the session timeout attribute specifies the maximum number of seconds the authenticator will wait before re-authenticating the client. A session timeout value of zero indicates another authentication request (possibly of a different type) immediately after the first authentication request has successfully completed.
- When specified in a RADIUS access-challenge message, the session timeout value represents the maximum number of seconds the Foundry device will wait for an EAP response from an 802.1X-enabled client before retransmitting the EAP-Request frame to the client.

NOTE: Together with the *termination-action* attribute, these commands can be used to provide quarantined VLAN remediation support.

Configuring Dynamic VLAN Assignment for 802.1X Ports

When a client/supplicant successfully completes the EAP authentication process, the Authentication Server (the RADIUS server) sends the Authenticator (the Foundry device) a RADIUS Access-Accept message that grants the client access to the network. The RADIUS Access-Accept message contains attributes set for the user in the user's access profile on the RADIUS server.

If one of the attributes in the Access-Accept message specifies a VLAN identifier, and this VLAN is available on the Foundry device, the client's port is moved from its default VLAN to the specified VLAN. When the client disconnects from the network, the port is placed back in its default VLAN.

NOTE: This feature is supported on port-based VLANs only. This feature cannot be used to place an 802.1X-enabled port into a Layer 3 protocol VLAN.

To enable 802.1X VLAN ID support, you must add the following attributes to a user's profile on the RADIUS server:

Attribute Name	**Type**	**Value**
Tunnel-Type	064	13 (decimal) – VLAN
Tunnel-Medium-Type	065	6 (decimal) – 802
Tunnel-Private-Group-ID	081	<vlan-name> (string) – either the name or the number of a VLAN configured on the Foundry device.

The device reads the attributes as follows:

- If the Tunnel-Type or the Tunnel-Medium-Type attributes in the Access-Accept message do not have the values specified above, the Foundry device ignores the three Attribute-Value pairs. The client becomes authorized, but the client's port is not dynamically placed in a VLAN.
- If the Tunnel-Type or the Tunnel-Medium-Type attributes in the Access-Accept message do have the values specified above, but there is no value specified for the Tunnel-Private-Group-ID attribute, the client will not become authorized.
- When the Foundry device receives the value specified for the Tunnel-Private-Group-ID attribute, it checks whether the <vlan-name> string matches the name of a VLAN configured on the device. If there is a VLAN on the device whose name matches the <vlan-name> string, then the client's port is placed in the VLAN whose ID corresponds to the VLAN name.
- If the <vlan-name> string does not match the name of a VLAN, the Foundry device checks whether the string, when converted to a number, matches the ID of a VLAN configured on the device. If it does, then the client's port is placed in the VLAN with that ID.
- If the <vlan-name> string does not match either the name or the ID of a VLAN configured on the device, then the client will not become authorized.

The **show interface** command displays the VLAN to which an 802.1X-enabled port has been dynamically assigned, as well as the port from which it was moved (that is, the port's default VLAN). See "Displaying Dynamically Assigned VLAN Information" on page 5-33 for sample output indicating the port's dynamically assigned VLAN.

Dynamic Multiple VLAN Assignment for 802.1X Ports on the FES, FESX, FSX, and FWSX

NOTE: This section applies to releases 02.2.00 and later for the FESX and FWSX, releases 3.3.00 and later for the FES, and releases 02.3.01 and later for the FSX.

When you add attributes to a user's profile on the RADIUS server, the <vlan-name> value for the Tunnel-Private-Group-ID attribute can specify the name or number of one or more VLANs configured on the Foundry device.

For example, to specify one VLAN, configure the following for the <vlan-name> value in the Tunnel-Private-Group-ID attribute on the RADIUS server:

"10" or "marketing"

In this example, the port on which the Client is authenticated is assigned to VLAN 10 or the VLAN named "marketing". The VLAN to which the port is assigned must have previously been configured on the Foundry device.

To specify an untagged VLAN:

"U:10" or "U:marketing"

When the RADIUS server specifies an untagged VLAN ID, the port's default VLAN ID (or ***PVID***) is changed from the system DEFAULT-VLAN (VLAN 1) to the specified VLAN ID. The port transmits only untagged traffic on its PVID. In this example, the port's PVID is changed from VLAN 1 (the DEFAULT-VLAN) to VLAN 10 or the VLAN named "marketing".

The PVID for a port can be changed only once through RADIUS authentication. For example, if RADIUS authentication for a Client causes a port's PVID to be changed from 1 to 10, and then RADIUS authentication for another Client on the same port specifies that the port's PVID be moved to 20, then the second PVID assignment from the RADIUS server is ignored.

If the link goes down, or the dot1x-mac-session for the Client that caused the initial PVID assignment ages out, then the port reverts back to its original (non-RADIUS-specified) PVID, and subsequent RADIUS authentication can change the PVID assignment for the port.

If a port's PVID is assigned through the multi-device port authentication feature, and 802.1X authentication subsequently specifies a different PVID, then the PVID specified through 802.1X authentication overrides the PVID specified through multi-device port authentication.

To specify tagged VLANs:

"T:12;T:20" or "T:12;T:marketing"

In this example, the port is added to VLANs 12 and 20 or VLANs 12 and the VLAN named "marketing". When a tagged packet is authenticated, and a list of VLANs is specified on the RADIUS server for the MAC address, then the packet's tag must match one of the VLANs in the list in order for the Client to be successfully authenticated. If authentication is successful, then the port is added to all of the VLANs specified in the list.

Unlike with a RADIUS-specified untagged VLAN, if the dot1x-mac-session for the Client ages out, the port's membership in RADIUS-specified tagged VLANs is not changed. In addition, if multi-device port authentication specifies a different list of tagged VLANs, then the port is added to the specified list of VLANs. Membership in the VLANs specified through 802.1X authentication is not changed.

To specify an untagged VLAN and multiple tagged VLANs:

"U:10;T:12;T:marketing"

When the RADIUS server returns a value specifying both untagged and tagged VLAN IDs, the port becomes a dual-mode port, accepting and transmitting both tagged traffic and untagged traffic at the same time. A dual-mode port transmits only untagged traffic on its default VLAN (PVID) and only tagged traffic on all other VLANs.

In this example, the port's VLAN configuration is changed so that it transmits untagged traffic on VLAN 10, and transmits tagged traffic on VLAN 12 and the VLAN named "marketing".

For a configuration example, see "802.1X Authentication with Dynamic VLAN Assignment" on page 5-44.

Saving Dynamic VLAN Assignments to the Running-Config File

NOTE: This topic is for FESX and FWSX running releases 02.2.00 and later.

You can configure the Foundry device to save the RADIUS-specified VLAN assignments to the device's running-config file. The configuration syntax for this feature differs on BI/FI and FES devices than on FESX and FWSX devices. On FESX and FWSX devices, enter commands such as the following:

```
FESX424 router(config)# dot1x-enable
FESX424 router(config-dot1x)# save-dynamicvlan-to-config
```

Syntax: save-dynamicvlan-to-config

By default, the dynamic VLAN assignments are not saved to the running-config file. Entering the **show running-config** command does not display dynamic VLAN assignments, although they can be displayed with the s**how vlan** and **show authenticated-mac-address detail** commands.

NOTE: When this feature is enabled, issuing the command **write mem** will save any dynamic VLAN assignments to the startup configuration file.

Considerations for Dynamic VLAN Assignment in an 802.1X Mutiple-Host Configuration

NOTE: This topic applies to Enterprise release 07.8.00 and later, and release 02.1.00 and later for the BigIron MG8 and NetIron 40G.

The following considerations apply when a Client in a 802.1X multiple-host configuration is successfully authenticated, and the RADIUS Access-Accept message specifies a VLAN for the port:

- If the port is not already a member of a RADIUS-specified VLAN, and the RADIUS Access-Accept message specifies the name or ID of a valid VLAN on the Foundry device, then the port is placed in that VLAN.
- If the port is already a member of a RADIUS-specified VLAN, and the RADIUS Access-Accept message specifies the name or ID of a different VLAN, then it is considered an authentication failure. The port's VLAN membership is not changed.

- If the port is already a member of a RADIUS-specified VLAN, and the RADIUS Access-Accept message specifies the name or ID of that same VLAN, then traffic from the Client is forwarded normally.
- If the RADIUS Access-Accept message specifies the name or ID of a VLAN that does not exist on the Foundry device, then it is considered an authentication failure.
- If the port is a tagged or dual-mode port, and the RADIUS Access-Accept message specifies the name or ID of a valid VLAN on the Foundry device, then the port is placed in that VLAN. If the port is already a member of the RADIUS-specified VLAN, no further action is taken. Note that the Client's dot1x-mac-session is set to "access-is-allowed" for the RADIUS-specified VLAN only. If traffic from the Client's MAC address is received on any other VLAN, it is dropped.
- If the RADIUS Access-Accept message does not contain any VLAN information, the Client's dot1x-mac-session is set to "access-is-allowed". If the port is already in a RADIUS-specified VLAN, it remains in that VLAN.

Using Dynamic VLAN Assignment with the MAC Port Security Feature

The MAC port security feature can be configured on 802.1X-enabled ports. The MAC port security feature allows the Foundry device to learn a limited number of "secure" MAC addresses on an interface. The interface will forward only packets with source MAC addresses that match these secure addresses. If the interface receives a packet with a source MAC address that is different from any of the secure addresses, it is considered a security violation, and subsequent packets from the violating MAC address can be dropped, or the port can be disabled entirely.

If a port has been disabled due to a MAC port security violation, 802.1X clients attempting to connect over the port cannot be authorized. In addition, 802.1X clients connecting from non-secure MAC addresses cannot be authorized.

To use 802.1X dynamic VLAN assignment with the MAC port security feature on an interface, you must set the number of secure MAC addresses to two or more. For example:

```
BigIron(config)# int e 3/2
BigIron(config-if-e100-3/2)# port security
BigIron(config-port-security-e100-3/2)# maximum 2
BigIron(config-port-security-e100-3/2)# exit
```

See the "Using the MAC Port Security Feature" on page 6-1 for more information.

NOTE: There is small chance that an interface can be inadvertently disabled when both 802.1X (with dynamic VLAN assignment) and the MAC port security feature are enabled on the interface. When this happens, disable then re-enable the interface to bring the interface back up.

Dynamically Applying IP ACLs and MAC Filters to 802.1X Ports

On Devices Running Enterprise Software Release 07.8.01 and Later

In Enterprise software release 07.8.00 and earlier, dynamic IP ACL and MAC address filter assignment was not supported in an 802.1X multiple-host configuration, only for single host connected to the port. In an 802.1X multiple-host configuration, if a RADIUS server returned an Access-Accept message that specified an IP ACL or MAC address filter for the Client, these attributes were ignored.

Starting with Enterprise software release 07.8.01, dynamic IP ACL and MAC address filter assignment is now supported in an 802.1X multiple-host configuration. If there are multiple hosts connected to a single 802.1X-enabled port, RADIUS-specified IP ACLs and MAC filters can be applied to each host, independent of the other hosts connected to the port.

Flow-Based IP ACLs

In releases prior to 07.8.00, IP ACLs that were dynamically assigned using a RADIUS server were ***rule-based***, meaning that when the IP ACL was assigned, entries were immediately programmed into CAM. Starting with release 07.8.00, dynamically assigned IP ACLs are **flow-based**, meaning that entries are programmed into CAM after the flow is processed by the CPU. A flow is defined as traffic with a common source IP address, destination IP address, protocol, source port, and destination port.

When this feature is configured, any new flow received on an interface is sent to the CPU for processing. If there is an IP ACL to be applied to the flow, based on its 802.1x information (authentication status and MAC address), the CPU programs CAM entries to permit or deny the flow.

To use the flow-based ACL mechanism, you must enable flow-based ACLs on the 802.1X-enabled interfaces, and you must define a “placeholder” ACL to force a packet from each new flow to the CPU for processing. If you are using router code, and wish to filter traffic on a virtual routing interface (VE), you must enable traffic filtering on the VE. If you want to filter packets denied by ACLs in hardware, you must enable hardware filtering on the device.

Notes:

- Dynamically assigned outbound ACLs are supported in switch code only.
- Only one dynamically assigned MAC filter can be applied on a port at a time. This means that if an 802.1X-enabled port currently has a dynamically assigned MAC filter applied to it, and a host on the same port is subsequently authenticated, then any MAC filter information returned for the second host is ignored (although the second host is authenticated).

Configuring Dynamic ACL Assignment for an 802.1X Multiple-Host Configuration

To configure dynamic ACL assignment for an 802.1X multiple-host configuration, you perform the following tasks:

- Enable dynamic ACLs and MAC address filters for 802.1X multiple-host configurations on the Foundry device
- Enable flow-based ACLs on the 802.1X-enabled interfaces
- Configure a “placeholder” ACL so that the initial packets of a flow are sent to the CPU for processing
- Enable traffic filtering on a virtual routing interface (if necessary)
- Enable hardware filtering of denied packets (if necessary)

Enabling Dynamic ACLs and MAC Address Filters for 802.1X Multiple-Host Configurations

To globally enable dynamically assigned IP ACLs and MAC address filters for 802.1X multiple-host configurations, enter the following commands:

```
BigIron#(config) dot1x enable
BigIron#(config-dot1x)# multi-user-policy enable
```

Syntax: [no] multi-user-policy enable

Enabling Flow-Based ACLs on the 802.1X Interfaces

Since the ACLs used in a 802.1X multiple-host configuration are flow-based, you must enable flow-based ACLs on the device. For example, to do this on interface 3/11, enter the following commands:

```
BigIron#(config) interface e 3/11
BigIron#(config-if-e1000-3/11)# ip access-group flow-mode
```

Syntax: [no] ip access-group flow-mode

Configuring a Placeholder ACL

Since the dynamically assigned ACLs used in 802.1X multiple-host configuration are flow-based, a packet from each new flow must be sent to the CPU for processing. If there is an IP ACL to be applied to the flow, based on its 802.1x information (authentication status and MAC address), the CPU then programs CAM entries to permit or deny the flow.

To cause the device to send the initial packet in a flow to the CPU, you create a “placeholder” ACL and apply it to the interface. This placeholder ACL should specify a host that does not exist in the network, so that the placeholder ACL does not affect traffic from a real host.

For example, the following commands create an ACL that filters TCP, UDP and/or ICMP traffic and then apply the ACL to inbound and outbound traffic on an interface:

```
BigIron(config)# access-list 131 deny tcp host 1.1.1.1 any
BigIron(config)# access-list 131 deny udp host 1.1.1.1 any
BigIron(config)# access-list 131 deny icmp host 1.1.1.1 any
BigIron(config)# access-list 131 permit ip any any

BigIron(config) interface e 3/11
```

```
BigIron#(config-if-e1000-3/11)# ip access-group flow-mode
BigIron#(config-if-e1000-3/11)# ip access-group 131 in
BigIron#(config-if-e1000-3/11)# ip access-group 131 out
BigIron#(config-if-e1000-3/11)# exit
```

When the placeholder ACL is applied, any new IP traffic flow on interface 3/11 is directed to the CPU. If the source MAC address of the flow is already associated with a successfully authenticated 802.1X host that has a dynamically assigned IP ACL applied to it, then that dynamically assigned IP ACL is applied to the flow. Note that if there is a user-defined ACL for this MAC address, the placeholder ACL is ignored, and only the user-defined ACL is applied to the flow.

Filtering Traffic on a Virtual Routing Interface (VE)

If the ACL is to process traffic on a virtual routing interface (VE), you must enable traffic filtering on the VE.

By default, the Foundry device does not filter traffic that is switched from one port to another within the same VE, even if an ACL is applied to the interface. You can enable the device to filter switched traffic within a virtual routing interface. When you enable the filtering, the device uses the ACLs applied to inbound traffic to filter traffic received by a port from another port in the same virtual routing interface. This feature does not apply to ACLs applied to outbound traffic.

For example, the following commands enable traffic filtering on VE 1:

```
BigIron(config)# int ve 1
BigIron(config-vif-1)# ip access-group ve-traffic
```

Syntax: [no] ip access-group ve-traffic

Enabling Hardware Filtering of Denied Packets

To configure the device to filter denied packets in hardware, rather than using the CPU, enter the following command:

```
BigIron(config)# hw-drop-acl-denied-packet
```

Syntax: [no] hw-drop-acl-denied-packet

When you enable hardware filtering of denied packets, the CPU creates a CAM entry for the denied packet. Subsequent packets with the same address information are filtered using the CAM entry. The CAM entry ages out after two minutes if not used.

On BigIron MG8 and NetIron 40G Running Release 02.1.00 and Later

The following configuration consideration must be added when using this feature with the BigIron MG8 or NetIron 40G:

- MAC address filters cannot be configured on the same port with inbound ACLs.

The BigIron MG8 and NetIron 40G use information in the Filter ID and Vendor-Specific attributes as follows:

- The Filter-ID attribute can specify the number of an existing IP ACL or MAC address filter configured on the Foundry device. In this case, the IP ACL or MAC address filter with the specified number is applied to the port.
- The Vendor-Specific attribute can specify actual syntax for a Foundry IP ACL or MAC address filter, which is then applied to the authenticated port. This involves configuring ***per-user*** IP ACLs or MAC address filters.

On the BigIron MG8 and NetIron 40G software release 02.2.01 and later, flow-based, not rule-based ACLs can be applied to 802.1X-enabled ports.

On the FES Running Release 03.3.00 and Later

NOTE: This section applies to release 03.3.00 and later for the FastIron Edge Switch.

Dynamic IP ACL and MAC address filter assignment is supported in an 802.1X multiple-host configuration:

- If there are multiple hosts connected to a single 802.1X-enabled port, RADIUS-specified IP ACLs can be applied to each host, independent of the other hosts connected to the port.

- A single RADIUS-specified MAC address filter can be applied to a port in an 802.1X multiple-host configuration. This MAC address filter affects all of the hosts connected to the port.

On the FES, dynamically assigned IP ACLs are ***flow-based***, meaning that entries are programmed into hardware after the flow is processed by the CPU. A flow is defined as traffic with a common source IP address, destination IP address, protocol, source port, and destination port. New flows received on an interface are sent to the CPU for processing. If there is an IP ACL to be applied to the flow, based on its 802.1X information (authentication status and MAC address), the CPU programs hardware entries to permit or deny the flow.

Notes:

- Dynamically assigned outbound ACLs are supported on both switch code and router images.
- The RADIUS server can return an IP ACL, MAC address filter, or neither. However, an IP ACL and a MAC address filter cannot be applied to the port at the same time.
- Dynamic MAC address filter assignment is enabled by default when 802.1X port security is enabled on a port.
- On router images, an IP ACL cannot be dynamically assigned to a port that is a member of a virtual routing interface (VE). However, you can use the 802.1X dynamic VLAN assignment feature to assign the port to a VE, and then the VE's ACLs would be assigned to the Client's port.
- On router images, an IP ACL is valid only on router ports (that is, ports with at least one IP address).
- On router images, a MAC filter can be applied only on switch ports (that is, ports with no IP address).
- Only one dynamically assigned MAC filter can be applied on a port at a time. This means that if an 802.1X-enabled port currently has a dynamically assigned MAC filter applied to it, and a host on the same port is subsequently authenticated, then any MAC filter information returned for the second host is ignored (although the second host is authenticated).
- If you want to filter packets denied by ACLs in hardware, you must enable hardware filtering on the device with the **hw-drop-acl-denied-packet** command. When you enable hardware filtering of denied packets, the CPU creates a hardware entry for the denied packet. Subsequent packets with the same address information are filtered using the hardware entry. The hardware entry ages out after two minutes if not used.

On the FESX, FSX, and FWSX

NOTE: This section applies to the FESX and FWSX running release 02.2.00 or later, and the FSX running release 02.3.01 or later.

Foundry's 802.1X implementation supports dynamically applying an IP ACL or MAC address filter to a port, based on information received from an Authentication Server.

When a client/supplicant successfully completes the EAP authentication process, the Authentication Server (the RADIUS server) sends the Authenticator (the Foundry device) a RADIUS Access-Accept message that grants the client access to the network. The RADIUS Access-Accept message contains attributes set for the user in the user's access profile on the RADIUS server.

If the Access-Accept message contains Filter-ID (type 11) and/or Vendor-Specific (type 26) attributes, the Foundry device can use information in these attributes to apply an IP ACL or MAC address filter to the authenticated port. This IP ACL or MAC address filter applies to the port for as long as the client is connected to the network. When the client disconnects from the network, the IP ACL or MAC address filter is no longer applied to the port. If an IP ACL or MAC address filter had been applied to the port prior to 802.1X authentication, it is then re-applied to the port.

The Foundry device uses information in the Filter ID and Vendor-Specific attributes as follows:

- The Filter-ID attribute can specify the number of an existing IP ACL or MAC address filter configured on the Foundry device. In this case, the IP ACL or MAC address filter with the specified number is applied to the port.
- The Vendor-Specific attribute can specify actual syntax for a Foundry IP ACL or MAC address filter, which is then applied to the authenticated port. Configuring a Vendor-Specific attribute in this way allows you to create IP ACLs and MAC filters that apply to individual users; that is, ***per-user*** IP ACLs or MAC address filters.

Configuration Considerations

The following restrictions apply to dynamic IP ACLs or MAC address filters:

- The FESX, FWSX, and FastIron SuperX support inbound dynamic IP ACLs only. These devices do not support outbound dynamic ACLs.
- The FESX, FWSX, and FastIron SuperX support inbound Vendor-Specific attributes only. These devices do not support outbound Vendor-Specific attributes.
- At most, one IP ACL can be configured in the inbound direction on an interface.
- MAC address filters cannot be configured in the outbound direction on an interface.
- The FESX, FWSX, and FastIron SuperX do not support concurrent operation of MAC address filters and IP ACLS.

Disabling and Enabling Strict Security Mode for Dynamic Filter Assignment

By default, 802.1X dynamic filter assignment operates in ***strict security mode***. When strict security mode is enabled, 802.1X authentication for a port fails if the Filter-ID attribute contains invalid information, or if insufficient system resources are available to implement the per-user IP ACLs or MAC address filters specified in the Vendor-Specific attribute.

When strict security mode is enabled:

- If the Filter-ID attribute in the Access-Accept message contains a value that does not refer to an existing filter (that is, a MAC address filter or IP ACL configured on the device), then the port will not be authenticated, regardless of any other information in the message (for example, if the Tunnel-Private-Group-ID attribute specifies a VLAN to which to assign the port).
- If the Vendor-Specific attribute specifies the syntax for a filter, but there are insufficient system resources to implement the filter, then the port will not be authenticated.
- If the device does not have the system resources available to dynamically apply a filter to a port, then the port will not be authenticated.

NOTE: If the Access-Accept message contains values for both the Filter-ID and Vendor-Specific attributes, then the value in the Vendor-Specific attribute (the per-user filter) takes precedence.

Also, if authentication for a port fails because the Filter-ID attribute referred to a non-existent filter, or there were insufficient system resources to implement the filter, then a Syslog message is generated.

When strict security mode is disabled:

- If the Filter-ID attribute in the Access-Accept message contains a value that does not refer to an existing filter (that is, a MAC address filter or IP ACL configured on the device), then the port is still authenticated, but no filter is dynamically applied to it.
- If the Vendor-Specific attribute specifies the syntax for a filter, but there are insufficient system resources to implement the filter, then the port is still authenticated, but the filter specified in the Vendor-Specific attribute is not applied to the port.

By default, strict security mode is enabled for all 802.1X-enabled interfaces, but you can manually disable or enable it, either globally or for specific interfaces.

To disable strict security mode globally, enter the following commands:

```
FESX448 Switch(config)# dot1x-enable
FESX448 Switch(config-dot1x)# no global-filter-strict-security
```

After you have globally disabled strict security mode on the device, you can re-enable it by entering the following command:

```
FESX448 Switch(config-dot1x)# global-filter-strict-security
```

Syntax: [no] global-filter-strict-security

To disable strict security mode for a specific interface, enter commands such as the following:

```
FESX448 Switch(config)# interface e 1
FESX448 Switch(config-if-1)# no dot1x filter-strict-security
```

To re-enable strict security mode for an interface, enter the following command:

```
FESX448 Switch(config-if-1)# dot1x filter-strict-security
```

Syntax: [no] dot1x filter-strict-security

The output of the **show dot1x** and **show dot1x config** commands has been enhanced to indicate whether strict security mode is enabled or disabled globally and on an interface. See "Displaying the Status of Strict Security Mode on the FastIron Edge Switch X-Series and FastIron SuperX" on page 5-35.

Dynamically Applying Existing ACLs or MAC Address Filters

When a port is authenticated using 802.1X security, an IP ACL or MAC address filter that exists in the running-config on the Foundry device can be dynamically applied to the port. To do this, you configure the Filter-ID (type 11) attribute on the RADIUS server. The Filter-ID attribute specifies the name or number of the Foundry IP ACL or MAC address filter.

The following is the syntax for configuring the Filter-ID attribute to refer to a Foundry IP ACL or MAC address filter:

Value	Description
ip.<number>.in	Applies the specified numbered ACL to the 802.1X authenticated port in the inbound direction.
ip.<name>.in	Applies the specified named ACL to the 802.1X authenticated port in the inbound direction.
mac.<number>.in	Applies the specified numbered MAC address filter to the 802.1X authenticated port in the inbound direction.

The following table lists examples of values you can assign to the Filter-ID attribute on the RADIUS server to refer to IP ACLs and MAC address filters configured on a Foundry device.

Possible Values for the Filter ID attribute on the RADIUS server	ACL or MAC address filter configured on the Foundry device
ip.2.in	access-list 2 permit host 36.48.0.3 access-list 2 permit 36.0.0.0 0.255.255.255
ip.102.in	access-list 102 permit ip 36.0.0.0 0.255.255.255 any
ip.fdry_filter.in	ip access-list standard fdry_filter permit host 36.48.0.3
mac.2.in	mac filter 2 permit 3333.3333.3333 ffff.ffff.ffff any etype eq 0800
mac.2.in mac.3.in	mac filter 2 permit 3333.3333.3333 ffff.ffff.ffff any etype eq 0800 mac filter 3 permit 2222.2222.2222 ffff.ffff.ffff any etype eq 0800

Notes

- The <name> in the Filter ID attribute is case-sensitive.
- You can specify only numbered MAC address filters in the Filter ID attribute. Named MAC address filters are not supported.
- Dynamic ACL filters are supported only for the inbound direction. Dynamic outbound ACL filters are not

supported.

- MAC address filters are supported only for the inbound direction. Outbound MAC address filters are not supported.
- Dynamically assigned IP ACLs and MAC address filters are subject to the same configuration restrictions as non-dynamically assigned IP ACLs and MAC address filters. See the *Foundry Enterprise Configuration and Management Guide* for more information.
- Multiple IP ACLs and MAC address filters can be specified in the Filter ID attribute, allowing multiple filters to be simultaneously applied to an 802.1X authenticated port. Use commas, semicolons, or carriage returns to separate the filters (for example: ip.3.in,mac.2.in).

Configuring Per-User IP ACLs or MAC Address Filters

Per-user IP ACLs and MAC address filters make use of the Vendor-Specific (type 26) attribute to dynamically apply filters to ports. Defined in the Vendor-Specific attribute are Foundry ACL or MAC address filter statements. When the RADIUS server returns the Access-Accept message granting a client access to the network, the Foundry device reads the statements in the Vendor-Specific attribute and applies these IP ACLs or MAC address filters to the client's port. When the client disconnects from the network, the dynamically applied filters are no longer applied to the port. If any filters had been applied to the port previous to the client connecting, then those filters are reapplied to the port.

The following is the syntax for configuring the Foundry Vendor-Specific attribute with ACL or MAC address filter statements:

Value	Description
ipacl.e.in=<extended-acl-entries>	Applies the specified extended ACL entries to the 802.1X authenticated port in the inbound direction.
macfilter.in=<mac-filter-entries>	Applies the specified MAC address filter entries to the 802.1X authenticated port in the inbound direction.

The following table shows examples of IP ACLs and MAC address filters configured in the Foundry Vendor-Specific attribute on a RADIUS server. These IP ACLs and MAC address filters follow the same syntax as other Foundry ACLs and MAC address filters. See the *Foundry Enterprise Configuration and Management Guide* for information on syntax.

ACL or MAC address filter	Vendor-Specific attribute on RADIUS server
MAC address filter with one entry	macfilter.in= deny any any

MAC address filter with two entries	macfilter.in= permit 0000.0000.3333 ffff.ffff.0000 any,
	macfilter.in= permit 0000.0000.4444 ffff.ffff.0000 any

The RADIUS server allows one instance of the Vendor-Specific attribute to be sent in an Access-Accept message. However, the Vendor-Specific attribute can specify multiple IP ACLs or MAC address filters. You can use commas, semicolons, or carriage returns to separate the filters (for example: ipacl.e.in= permit ip any any,ipacl.e.in = deny ip any any).

Enabling 802.1X Port Security

By default, 802.1X port security is disabled on Foundry devices. To enable the feature on the device and enter the dot1x configuration level, enter the following command:

```
BigIron(config)# dot1x-enable
BigIron(config-dot1x)#
```

Syntax: [no] dot1x-enable

At the dot1x configuration level, you can enable 802.1X port security on all interfaces at once, on individual interfaces, or on a range of interfaces.

For example, to enable 802.1X port security on all interfaces on the device, enter the following command:

```
BigIron(config-dot1x)# enable all
```

Syntax: [no] enable all

To enable 802.1X port security on interface 3/11, enter the following command:

```
BigIron(config-dot1x)# enable ethernet 3/11
```

Syntax: [no] enable <portnum>

To enable 802.1X port security on interfaces 3/11 through 3/16, enter the following command:

```
BigIron(config-dot1x)# enable ethernet 3/11 to 3/16
```

Syntax: [no] enable <portnum> to <portnum>

Setting the Port Control

To activate authentication on an 802.1X-enabled interface, you specify the kind of ***port control*** to be used on the interface. An interface used with 802.1X port security has two virtual access points: a controlled port and an uncontrolled port.

- The controlled port can be either the authorized or unauthorized state. In the authorized state, it allows normal traffic to pass between the Client and the Authenticator. In the unauthorized state, it allows no traffic to pass through.
- The uncontrolled port allows only EAPOL traffic between the Client and the Authentication Server.

See Figure 5.3 on page 5-4 for an illustration of this concept.

By default, all controlled ports on the device are in the authorized state, allowing all traffic. When you activate authentication on an 802.1X-enabled interface, its controlled port is placed in the unauthorized state. When a Client connected to the interface is successfully authenticated, the controlled port is then placed in the authorized state. The controlled port remains in the authorized state until the Client logs off.

To activate authentication on an 802.1X-enabled interface, you configure the interface to place its controlled port in the authorized state when a Client is authenticated by an Authentication Server. To do this, enter commands such as the following.

```
BigIron(config)# interface e 3/1
BigIron(config-if-3/1)# dot1x port-control auto
```

Syntax: [no] dot1x port-control [force-authorized | force-unauthorized | auto]

When an interface's control type is set to **auto**, the its controlled port is initially set to unauthorized, but is changed to authorized when the connecting Client is successfully authenticated by an Authentication Server.

The port control type can be one of the following:

force-authorized – The port's controlled port is placed unconditionally in the authorized state, allowing all traffic. This is the default state for ports on the Foundry device.

force-unauthorized – The controlled port is placed unconditionally in the unauthorized state.

auto – The controlled port is unauthorized until authentication takes place between the Client and Authentication Server. Once the Client passes authentication, the port becomes authorized. This has the effect of activating authentication on an 802.1X-enabled interface.

NOTE: You cannot enable 802.1X port security on ports that have any of the following features enabled:

- 10 Gbps ports (only for BigIron MG8 and NetIron 40G software release 02.1.00 and later)
- Static MAC configurations (only for BigIron MG8 and NetIron 40G software release 02.1.00 and later)
- Link aggregation
- Metro Ring Protocol (MRP)
- Tagged port
- Mirror port
- Trunk port

In releases prior to 07.6.03, 802.1X port security could not be enabled on a port where Layer 2 switching was disabled (with the **route-only** command), and an 802.1X port could not be specified as a member of a virtual interface (ve). Both of these restrictions were removed in release 07.6.03.

Configuring Periodic Re-Authentication

You can configure the device to periodically re-authenticate Clients connected to 802.1X-enabled interfaces. When you enable periodic re-authentication, the device re-authenticates Clients every 3,600 seconds by default. You can optionally specify a different re-authentication interval of between 1 – 4294967295 seconds.

To configure periodic re-authentication using the default interval of 3,600 seconds, enter the following command:

```
BigIron(config-dot1x)# re-authentication
```

Syntax: [no] re-authentication

To configure periodic re-authentication with an interval of 2,000 seconds, enter the following commands:

```
BigIron(config-dot1x)# re-authentication
BigIron(config-dot1x)# timeout re-authperiod 2000
```

Syntax: [no] timeout re-authperiod <seconds>

The re-authentication interval is a global setting, applicable to all 802.1X-enabled interfaces. If you want to re-authenticate Clients connected to a specific port manually, use the **dot1x re-authenticate** command. See “Re-Authenticating a Port Manually”, below.

Re-Authenticating a Port Manually

When periodic re-authentication is enabled, by default the Foundry device re-authenticates Clients connected to an 802.1X-enabled interface every 3,600 seconds (or the time specified by the **dot1x timeout re-authperiod** command). You can also manually re-authenticate Clients connected to a specific port.

For example, to re-authenticate Clients connected to interface 3/1, enter the following command:

```
BigIron# dot1x re-authenticate e 3/1
```

Syntax: dot1x re-authenticate <portnum>

Specifying the Security Hold Time

NOTE: This feature is not supported on BigIron MG8, NetIron 40G, or FastIron X-Series devices.

The **multiple-hosts** command (see “Allowing Access to Multiple Hosts” on page 5-26) allows more than one 802.1X Client to connect on an interface. However, when the **multiple-hosts** command is not used in an interface’s configuration, only one Client can connect on the interface. If the Foundry device detects multiple Clients trying to connect on an interface when the **multiple-hosts** command is not present in the interface’s configuration, the interface enters the unauthorized state for a specified amount of time. This amount of time is specified with the **security-hold-time** parameter. The **security-hold-time** parameter can be from 1 – 4294967295 seconds. The default is 60 seconds.

For example, the following command causes the device to place an interface in the unauthorized state for 120 seconds when it detects more than one 802.1X Client trying to connect on the interface:

```
BigIron(config-dot1x)# timeout security-hold-time 120
```

Syntax: [no] timeout security-hold-time <seconds>

NOTE: When the **port-control** parameter on an 802.1X-enabled interface is set to **force-authorized**, the Foundry device allows connections from multiple Clients, regardless of whether the **multiple-hosts** parameter is used in the interface's configuration.

Setting the Quiet Period

If the Foundry device is unable to authenticate the Client, the Foundry device waits a specified amount of time before trying again. The amount of time the Foundry device waits is specified with the **quiet-period** parameter. The **quiet-period** parameter can be from 0 – 4294967295 seconds. The default is 60 seconds.

For example, to set the quiet period to 30 seconds, enter the following command:

```
BigIron(config-dot1x)# timeout quiet-period 30
```

Syntax: [no] timeout quiet-period <seconds>

Specifying the Wait Interval and Number of EAP-Request/Identity Frame Retransmissions from the Foundry Device

When the Foundry device sends an EAP-request/identity frame to a Client, it expects to receive an EAP-response/identity frame from the Client. By default, if the Foundry device does not receive an EAP-response/identity frame from a Client, the device waits 30 seconds, then retransmits the EAP-request/identity frame. Also by default, the Foundry device retransmits the EAP-request/identity frame a maximum of two times. You can optionally configure the amount of time the device will wait before retransmitting an EAP-request/identity frame, and the number of times the EAP-request/identity frame will be transmitted. This section provides the command syntax for these features.

Setting the Wait Interval for EAP Frame Retransmissions

By default, if the Foundry device does not receive an EAP-response/identity frame from a Client, the device waits 30 seconds, then retransmits the EAP-request/identity frame. You can optionally change the amount of time the Foundry device waits before retransmitting the EAP-request/identity frame to the Client.

For example, to cause the Foundry device to wait 60 seconds before retransmitting an EAP-request/identity frame to a Client, enter the following command:

```
BigIron(config-dot1x)# timeout tx-period 60
```

If the Client does not send back an EAP-response/identity frame within 60 seconds, the device will transmit another EAP-request/identity frame.

Syntax: [no] timeout tx-period <seconds>

where <seconds> is a value from 0 – 4294967295. The default is 30 seconds.

Setting the Maximum Number of EAP Frame Retransmissions

The Foundry device retransmits the EAP-request/identity frame a maximum of two times. If no EAP-response/identity frame is received from the Client after two EAP-request/identity frame retransmissions (or the amount of time specified with the **auth-max** command), the device restarts the authentication process with the Client.

You can optionally change the number of times the Foundry device should retransmit the EAP-request/identity frame. You can specify between 1 – 10 frame retransmissions. For example, to configure the device to retransmit an EAP-request/identity frame to a Client a maximum of three times, enter the following command:

```
BigIron(config-dot1x)# maxreq 3
```

Syntax: maxreq <value>

where <value> is a number from 1 – 10. The default is 2.

For the FESX and FWSX devices running releases 02.2.00 and later and the FSX devices running 02.3.01 and later, use the **auth-max** command in lieu of the **maxreq** command, such as the following:

```
FESX424 router(config-dot1x)# auth-max 3
```

Syntax: auth-max <value>

where <value> is a number from 1 – 10. The default is 2.

Specifying the Wait Interval and Number of EAP-Request/Identity Frame Retransmissions from the RADIUS Server

Acting as an intermediary between the RADIUS Authentication Server and the Client, the Foundry device receives RADIUS messages from the RADIUS server, encapsulates them as EAPOL frames, and sends them to the Client. By default, when the Foundry device relays an EAP-Request frame from the RADIUS server to the Client, it expects to receive a response from the Client within 30 seconds. If the Client does not respond within the allotted time, the device retransmits the EAP-Request frame to the Client. Also by default, the Foundry device retransmits the EAP-request frame twice. If no EAP-response frame is received from the Client after two EAP-request frame retransmissions, the device restarts the authentication process with the Client.

You can optionally configure the amount of time the device will wait before retransmitting an EAP-request/identity frame, and the number of times the EAP-request/identity frame will be transmitted. This section provides the command syntax for these features.

Setting the Wait Interval for EAP Frame Retransmissions

By default, when the Foundry device relays an EAP-Request frame from the RADIUS server to the Client, it expects to receive a response from the Client within 30 seconds. You can optionally specify the wait interval using the **supptimeout** command.

For example, to configure the device to retransmit an EAP-Request frame if the Client does not respond within 45 seconds, enter the following command:

```
BigIron(config-dot1x)# supptimeout 45
```

Syntax: supptimeout <seconds>

where <seconds> is a number from 0 – 4294967295 seconds. The default is 30 seconds.

Setting the Maximum Number of EAP Frame Retransmissions

You can optionally specify the number of times the Foundry device will retransmit the EAP-request frame. You can specify between 1 – 10 frame retransmissions. For example, to configure the device to retransmit an EAP-request frame to a Client a maximum of three times, enter the following command:

```
BigIron(config-dot1x)# max-req 3
```

Syntax: max-req <value>

where <value> is a number from 1 – 10. The default is 2.

Specifying a Timeout for Retransmission of Messages to the Authentication Server

When performing authentication, the Foundry device receives EAPOL frames from the Client and passes the messages on to the RADIUS server. The device expects a response from the RADIUS server within 30 seconds. If the RADIUS server does not send a response within 30 seconds, the Foundry device retransmits the message to the RADIUS server. The time constraint for retransmission of messages to the Authentication Server can be between 0 – 4294967295 seconds.

For the BigIron MG8, the possible values are: 1 - 4294967295.

For example, to configure the device to retransmit a message if the Authentication Server does not respond within 45 seconds, enter the following command:

```
BigIron(config-dot1x)# servertimeout 45
```

Syntax: servertimeout <seconds>

Initializing 802.1X on a Port

To initialize 802.1X port security on a port, enter a command such as the following:

```
BigIron# dot1x initialize e 3/1
```

Syntax: dot1x initialize <portnum>

Allowing Access to Multiple Hosts

Foundry devices support 802.1X authentication for ports with more than one host connected to them. Multiple-host authentication works differently according to the software release running on the Foundry device:

- In releases prior to 07.8.00, services are provided on a port based on the authentication of a single Client. When one Client is successfully authenticated, all hosts connected to the port are allowed access to the network. See "Configuring 802.1X Multiple-Host Authentication (Enterprise Software Releases Prior to 07.8.00 only)".
- Starting in release 07.8.00, if there are multiple hosts connected to a single 802.1X-enabled port, the Foundry device authenticates each of them individually. See "Configuring 802.1X Multiple-Host Authentication (Enterprise Software Release 07.8.00 and Later and BigIron MG8 and NetIron 40G Software Release 02.1.00 and Later)".

Configuring 802.1X Multiple-Host Authentication (Enterprise Software Releases Prior to 07.8.00 only)

To enable 802.1X port security in a multiple-host configuration, a Foundry device running a release prior to 07.8.00 must be configured to allow multiple Clients on the same port. When one Client is successfully authenticated, all Clients connected to the port are allowed access to the network. When the authenticated Client logs off the network, the port becomes unauthorized again. Each time an authenticated Client logs off, the port becomes unauthorized.

To allow multiple 802.1X Clients on the same port, enter the following command:

```
BigIron(config-if-3/1)# dot1x multiple-hosts
```

Syntax: [no] dot1x multiple-hosts

By default multiple-host access is disabled. See Figure 5.7 on page 5-43 for a sample 802.1X configuration with multiple hosts connected to one port.

NOTE: When the **port-control** parameter on an 802.1X-enabled interface is set to **force-authorized**, the Foundry device allows connections from multiple Clients, regardless of whether the **multiple-hosts** parameter is used in the interface's configuration.

Configuring 802.1X Multiple-Host Authentication (Enterprise Software Release 07.8.00 and Later and BigIron MG8 and NetIron 40G Software Release 02.1.00 and Later)

When multiple hosts are connected to the same 802.1X-enabled port, the functionality described in "How 802.1X Multiple-Host Authentication Works" on page 5-7 is enabled by default. You can optionally do the following:

- Specify the authentication-failure action
- Specify the number of authentication attempts the device makes before dropping packets
- Disabling aging for dot1x-mac-sessions
- Configure aging time for blocked Clients
- Clear the dot1x-mac-session for a MAC address

Specifying the Authentication-Failure Action

In an 802.1X multiple-host configuration, if RADIUS authentication for a Client is unsuccessful, traffic from that Client is either dropped in hardware (the default), or the Client's port is placed in a "restricted" VLAN. You can specify which of these two authentication-failure actions is to be used. If the authentication-failure action is to place the port in a restricted VLAN, you can specify the ID of the restricted VLAN.

To specify that the authentication-failure action is to place the Client's port in a restricted VLAN, enter the following command:

```
BigIron(config)# dot1x-enable
BigIron(config-dot1x)# auth-fail-action restricted-vlan
```

Syntax: [no] auth-fail-action restricted-vlan

To specify the ID of the restricted VLAN as VLAN 300, enter the following command:

```
BigIron(config-dot1x)# auth-fail-vlanid 300
```

Syntax: [no] auth-fail-vlanid <vlan-id>

Specifying the Number of Authentication Attempts the Device Makes Before Dropping Packets

When the authentication-failure action is to drop traffic from the Client, and the initial authentication attempt made by the device to authenticate the Client is unsuccessful, the Foundry device immediately retries to authenticate the Client. After three unsuccessful authentication attempts, the Client's dot1x-mac-session is set to "access-denied", causing traffic from the Client to be dropped in hardware.

You can optionally configure the number of authentication attempts the device makes before dropping traffic from the Client. To do so, enter a command such as the following:

```
BigIron(config-dot1x)# auth-fail-max-attempts 2
```

Syntax: [no] auth-fail-max-attempts <attempts>

By default, the device makes 3 attempts to authenticate a Client before dropping packets from the Client. You can specify between 1 – 10 authentication attempts.

Disabling Aging for dot1x-mac-sessions

The dot1x-mac-sessions for Clients authenticated or denied by a RADIUS server are aged out if no traffic is received from the Client's MAC address for a certain period of time. After a Client's dot1x-mac-session is aged out, the Client must be re-authenticated.

- ***Permitted*** dot1x-mac-sessions, which are the dot1x-mac-sessions for authenticated Clients, as well as for non-authenticated Clients whose ports have been placed in the restricted VLAN, are aged out if no traffic is received from the Client's MAC address over the Foundry device's normal MAC aging interval.
- ***Denied*** dot1x-mac-sessions, which are the dot1x-mac-sessions for non-authenticated Clients that are blocked by the Foundry device are aged out over a configurable software aging period. (See the next section for more information on configuring the software aging period).

You can optionally disable aging of the permitted and/or denied dot1x-mac-sessions on the Foundry device.

To disable aging of the permitted dot1x-mac-sessions, enter the following command:

```
BigIron(config-dot1x)# mac-session-aging no-aging permitted-mac-only
```

Syntax: [no] mac-session-aging no-aging permitted-mac-only

To disable aging of the denied dot1x-mac-sessions, enter the following command:

```
BigIron(config-dot1x)# mac-session-aging no-aging denied-mac-only
```

Syntax: [no] mac-session-aging no-aging denied-mac-only

NOTE: This command enables aging of permitted sessions.

As a shortcut, you can use the command **[no] mac-session-aging** to enable or disable aging for both permitted and denied sessions.

Specifying the Aging Time for Blocked Clients

When the Foundry device is configured to drop traffic from non-authenticated Clients, traffic from the blocked Clients is dropped in hardware, without being sent to the CPU. A Layer 2 CAM entry is created that drops traffic from the blocked Client's MAC address in hardware. If no traffic is received from the blocked Client's MAC address for a certain amount of time, this Layer 2 CAM entry is aged out. If traffic is subsequently received from the Client's MAC address, then an attempt can be made to authenticate the Client again.

Aging of the Layer 2 CAM entry for a blocked Client's MAC address occurs in two phases, known as ***hardware aging*** and ***software aging***. The hardware aging period is fixed at 70 seconds and is non-configurable. The software aging time is configurable through the CLI.

Once the Foundry device stops receiving traffic from a blocked Client's MAC address, the hardware aging begins and lasts for a fixed period of time. After the hardware aging period ends, the software aging period begins. The software aging period lasts for a configurable amount of time (by default 120 seconds). After the software aging period ends, the blocked Client's MAC address ages out, and can be authenticated again if the Foundry device receives traffic from the Client's MAC address.

To change the length of the software aging period for a blocked Client's MAC address, enter a command such as the following:

```
BigIron(config)# mac-session-aging max-age 180
```

Syntax: [no] mac-session-aging max-age <seconds>

You can specify from 1 – 65535 seconds. The default is 120 seconds.

Clearing a dot1x-mac-session for a MAC Address

You can clear the dot1x-mac-session for a specified MAC address, so that the Client with that MAC address can be re-authenticated by the RADIUS server. For example:

```
BigIron# clear dot1x mac-session 00e0.1234.abd4
```

Syntax: clear dot1x mac-session <mac-address>

Defining MAC Filters for EAP Frames

You can create MAC address filters to permit or deny EAP frames. To do this, you specify the Foundry device's 802.1X group MAC address as the destination address in a MAC filter, then apply the filter to an interface.

MAC Filters for EAPS on most devices

For example, the following command creates a MAC filter that denies frames with the destination MAC address of 0180.c200.0003, which is the Foundry device's 802.1X group MAC address:

```
BigIron(config)# mac filter 1 deny any 0180.c200.0003 ffff.ffff.ffff
```

The following commands apply this filter to interface e 3/1:

```
BigIron(config)# interface e 3/11
BigIron(config-if-3/1)# mac filter-group 1
```

See "Defining MAC Address Filters" in the *Foundry Switch and Router Installation and Basic Configuration Guide* for more information.

MAC Filters for EAPs on the BigIron MG8 and NetIron 40G

Instead of using the **mac filter-group** command to define MAC filters for EAP frames, the BigIron MG8 and NetIron 40G use the **mac access-group** command. For example

```
BigIron MG8(config)# int e 4/12
BigIron MG8(config-int-e100-4/12)# mac access-group 400 in
```

Syntax: [no] mac access-group <num> in

The <num> parameter specifies the Layer 2 ACL table ID to bind to the interface.

See the "Layer 2 ACLs" chapter in the *Foundry Enterprise Configuration and Management Guide*.

Configuring Guest VLAN Access for Non-EAP-Capable Clients

NOTE: This feature is supported in releases 02.2.00 and later for the FESX and FWSX.

You can configure the Foundry device to grant "guest" VLAN access to clients that do not support Extensible Authentication Protocol (EAP). The guest VLAN (also called restricted VLAN) limits access to the network or applications, instead of blocking access to these services altogether.

When the Foundry device receives the first packet (non-EAP packet) from a client, the device waits for 10 seconds or the amount of time specified with the **timeout restrict-fwd-period** command. If the Foundry device does not receive subsequent packets after the timeout period, the device places the client onto the guest VLAN.

This feature is disabled by default. To enable this feature and change the timeout period, enter commands such as the following:

```
FESX424 router(config)# dot1x-enable
FESX424 router(config-dot1x)# restrict-forward-non-dot1x
FESX424 router(config-dot1x)# timeout restrict-fwd-period 15
```

Syntax: restrict-forward-non-dot1x

Syntax: timeout restrict-fwd-period <num>

The <num> parameter is a value from 0 to 4294967295. The default value is 10.

Displaying 802.1X Information

You can display the following 802.1X-related information:

- Information about the 802.1X configuration on the device and on individual ports
- Statistics about the EAPOL frames passing through the device
- Information about 802.1X-enabled ports dynamically assigned to a VLAN
- Information about the user-defined and dynamically applied MAC filters and IP ACLs currently active on the device
- Information about the 802.1X multiple-host configuration

Displaying 802.1X Configuration Information

To display information about the 802.1X configuration on the Foundry device, enter the following command:

```
BigIron# show dot1x
PAE Capability:    Authenticator Only
system-auth-control: Enable
re-authentication: Disable
global-filter-strict-security: Enable
quiet-period:    60 Seconds
tx-period:    30 Seconds
supptimeout:    30 Seconds
servertimeout:     30 Seconds
maxreq:    2
re-authperiod:    3600 Seconds
security-hold-time: 60 Seconds
Protocol Version:    1
```

Syntax: show dot1x

The following table describes the information displayed by the **show dot1x** command.

Table 5.1: Output from the show dot1x command

This Field...	Displays...
PAE Capability	The Port Access Entity (PAE) role for the Foundry device. This is always “Authenticator Only”.
system-auth-control	Whether system authentication control is enabled on the device. The **dot1x-enable** command enables system authentication control on the device.
re-authentication	Whether periodic re-authentication is enabled on the device. See “Configuring Periodic Re-Authentication” on page 5-23. When periodic re-authentication is enabled, the device automatically re-authenticates Clients every 3,600 seconds by default.
quiet-period	When the Foundry device is unable to authenticate a Client, the amount of time the Foundry device waits before trying again (default 60 seconds). See “Setting the Quiet Period” on page 5-24 for information on how to change this setting.
tx-period	When a Client does not send back an EAP-response/identity frame, the amount of time the Foundry device waits before retransmitting the EAP-request/identity frame to a Client (default 30 seconds). See “Setting the Wait Interval for EAP Frame Retransmissions” on page 5-24 for information on how to change this setting.
supp-timeout	When a Client does not respond to an EAP-request frame, the amount of time before the Foundry device retransmits the frame. See “Setting the Wait Interval for EAP Frame Retransmissions” on page 5-25 for information on how to change this setting.
server-timeout	When the Authentication Server does not respond to a message sent from the Client, the amount of time before the Foundry device retransmits the message. See “Specifying a Timeout for Retransmission of Messages to the Authentication Server” on page 5-25 for information on how to change this setting.
max-req	The number of times the Foundry device retransmits an EAP-request/ identity frame if it does not receive an EAP-response/identity frame from a Client (default 2 times). See “Setting the Maximum Number of EAP Frame Retransmissions” on page 5-24 for information on how to change this setting.
re-authperiod	How often the device automatically re-authenticates Clients when periodic re-authentication is enabled (default 3,600 seconds). See “Configuring Periodic Re-Authentication” on page 5-23 for information on how to change this setting.

Table 5.1: Output from the show dot1x command (Continued)

This Field...	Displays...
security-hold-time	The amount of time the device disables an interface when it detects multiple Clients trying to connect on the interface, when the **multiple-hosts** command is not present in the interface's configuration. See "Specifying the Security Hold Time" on page 5-23 for information on how to change this setting.
Protocol Version	The version of the 802.1X protocol in use on the device.

To display information about the 802.1X configuration on an individual port, enter a command such as the following:

```
BigIron# show dot1x config e 1/3

Port 1/3 Configuration:
Authenticator PAE state:    AUTHENTICATED
Backend Authentication state:     IDLE
AdminControlledDirections:    BOTH
OperControlledDirections:    BOTH
AuthControlledPortControl:     Auto
AuthControlledPortStatus:    authorized
quiet-period:    60 Seconds
tx-period:    30 Seconds
supptimeout:    30 Seconds
servertimeout:     30 Seconds
maxreq:    2
re-authperiod:    3600 Seconds
security-hold-time: 60 Seconds
re-authentication: Disable
multiple-hosts: Disable
filter-strict-security: Enable
Protocol Version:    1
```

Syntax: show dot1x config <portnum>

The following additional information is displayed in the **show dot1x config** command for an interface:

Table 5.2: Output from the show dot1x config command for an interface

This Field...	Displays...
Authenticator PAE state	The current status of the Authenticator PAE state machine. This can be INITIALIZE, DISCONNECTED, CONNECTING, AUTHENTICATING, AUTHENTICATED, ABORTING, HELD, FORCE_AUTH, or FORCE_UNAUTH. **Note:** When the Authenticator PAE state machine is in the AUTHENTICATING state, if the reAuthenticate, eapStart, eapLogoff, or authTimeout parameters are set to TRUE, it may place the Authenticator PAE state machine indefinitely in the ABORTING state. If this should happen, use the **dot1x initialize** command to initialize 802.1X port security on the port, or unplug the Client or hub connected to the port, then reconnect it.

Table 5.2: Output from the show dot1x config command for an interface (Continued)

This Field...	Displays...
Backend Authentication state	The current status of the Backend Authentication state machine. This can be REQUEST, RESPONSE, SUCCESS, FAIL, TIMEOUT, IDLE, or INITIALIZE.
AdminControlledDirections	Indicates whether an unauthorized controlled port exerts control over communication in both directions (disabling both reception of incoming frames and transmission of outgoing frames), or just in the incoming direction (disabling only reception of incoming frames). On Foundry devices, this parameter is set to BOTH.
OperControlledDirections	The setting for the OperControlledDirections parameter, as defined in the 802.1X standard. According to the 802.1X standard, if the AdminControlledDirections parameter is set to BOTH, the OperControlledDirections parameter is unconditionally set to BOTH. Since the AdminControlledDirections parameter on Foundry devices is always set to BOTH, the OperControlledDirections parameter is also set to BOTH.
AuthControlledPortControl	The port control type configured for the interface. If set to auto, authentication is activated on the 802.1X-enabled interface.
AuthControlledPortStatus	The current status of the interface's controlled port: either authorized or unauthorized.
multiple-hosts	Whether the port is configured to allow multiple Supplicants accessing the interface on the Foundry device through a hub. See "Allowing Access to Multiple Hosts" on page 5-26 for information on how to change this setting.

Displaying 802.1X Statistics

To display 802.1X statistics for an individual port, enter a command such as the following:

```
BigIron# show dot1x statistics e 3/3

Port 3/3 Statistics:
RX EAPOL Start:    0
RX EAPOL Logoff:    0
RX EAPOL Invalid:    0
RX EAPOL Total:    0
RX EAP Resp/Id:    0
RX EAP Resp other than Resp/Id:    0
RX EAP Length Error:    0
Last EAPOL Version:    0
Last EAPOL Source:    0007.9550.0B83
TX EAPOL Total:    217
TX EAP Req/Id:    163
TX EAP Req other than Req/Id:    0
```

Syntax: show dot1x statistics <portnum>

The following table describes the information displayed by the **show dot1x statistics** command for an interface.

Table 5.3: Output from the show dot1x statistics command

This Field...	Displays...
RX EAPOL Start	The number of EAPOL-Start frames received on the port.
RX EAPOL Logoff	The number of EAPOL-Logoff frames received on the port.
RX EAPOL Invalid	The number of invalid EAPOL frames received on the port.
RX EAPOL Total	The total number of EAPOL frames received on the port.
RX EAP Resp/Id	The number of EAP-Response/Identity frames received on the port
RX EAP Resp other than Resp/Id	The total number of EAPOL-Response frames received on the port that were not EAP-Response/Identity frames.
RX EAP Length Error	The number of EAPOL frames received on the port that have an invalid packet body length.
Last EAPOL Version	The version number of the last EAPOL frame received on the port.
Last EAPOL Source	The source MAC address in the last EAPOL frame received on the port.
TX EAPOL Total	The total number of EAPOL frames transmitted on the port.
TX EAP Req/Id	The number of EAP-Request/Identity frames transmitted on the port.
TX EAP Req other than Req/Id	The number of EAP-Request frames transmitted on the port that were not EAP-Request/Identity frames.

Clearing 802.1X Statistics

You can clear the 802.1X statistics counters on all interfaces at once, on individual interfaces, or on a range of interfaces.

For example, to clear the 802.1X statistics counters on all interfaces on the device, enter the following command:

```
BigIron# clear dot1x statistics all
```

Syntax: clear dot1x statistics all

To clear the 802.1X statistics counters on interface e 3/11, enter the following command:

```
BigIron# clear dot1x statistics e 3/11
```

Syntax: clear dot1x statistics <portnum>

Displaying Dynamically Assigned VLAN Information

The **show interface** command displays the VLAN to which an 802.1X-enabled port has been dynamically assigned, as well as the port from which it was moved (that is, the port's default VLAN).

The following is an example of the **show interface** command indicating the port's dynamically assigned VLAN. Information about the dynamically assigned VLAN is shown in bold type.

```
BigIron# show interface e 12/2
FastEthernet12/2 is up, line protocol is up
  Hardware is FastEthernet, address is 0204.80a0.4681 (bia 0204.80a0.4681)
  Configured speed auto, actual 100Mbit, configured duplex fdx, actual fdx
  Member of L2 VLAN ID 2 (dot1x-RADIUS assigned), original L2 VLAN ID is 1,
  port is untagged, port state is FORWARDING
  STP configured to ON, priority is level0, flow control enabled
  mirror disabled, monitor disabled
  Not member of any active trunks
  Not member of any configured trunks
  No port name
  MTU 1518 bytes
  300 second input rate: 0 bits/sec, 0 packets/sec, 0.00% utilization
  300 second output rate: 256 bits/sec, 0 packets/sec, 0.00% utilization
  3 packets input, 192 bytes, 0 no buffer
  Received 0 broadcasts, 0 multicasts, 3 unicasts
  0 input errors, 0 CRC, 0 frame, 0 ignored
  0 runts, 0 giants, DMA received 3 packets
  919 packets output, 58816 bytes, 0 underruns
  Transmitted 1 broadcasts, 916 multicasts, 2 unicasts
  0 output errors, 0 collisions, DMA transmitted 919 packets
```

In this example, the 802.1X-enabled port has been moved from VLAN 1 to VLAN 2. When the client disconnects, the port will be moved back to VLAN 1.

The **show run** command also indicates the VLAN to which the port has been dynamically assigned. The output can differ depending on whether GARP VLAN Registration Protocol (GVRP) is enabled on the device:

- **Without GVRP** – When you enter the **show run** command, the output indicates that the port is a member of the VLAN to which it was dynamically assigned through 802.1X. If you then enter the **write memory** command, the VLAN to which the port is currently assigned becomes the port's default VLAN in the device's configuration.
- **With GVRP** – When you enter the **show run** command, if the VLAN name supplied by the RADIUS server corresponds to a VLAN learned through GVRP, then the output indicates that the port is a member of the VLAN to which it was originally assigned (not the VLAN to which it was dynamically assigned).

  If the VLAN name supplied by the RADIUS server corresponds to a statically configured VLAN, the output indicates that the port is a member of the VLAN to which it was dynamically assigned through 802.1X. If you then enter the **write memory** command, the VLAN to which the port is currently assigned becomes the port's default VLAN in the device's configuration.

Displaying Information About Dynamically Applied MAC Filters and IP ACLs

You can display information about the user-defined and dynamically applied MAC filters and IP ACLs currently active on the device.

Displaying User-Defined MAC Filters and IP ACLs

To display the user-defined MAC filters active on the device, enter the following command:

```
BigIron# show dot1x mac-address-filter

Port 1/3 (User defined MAC Address Filter):
    mac filter 1 permit any any
```

Syntax: show dot1x mac-address-filter

To display the user-defined IP ACLs active on the device, enter the following command:

```
BigIron# show dot1x ip-acl

Port 1/3 (User defined IP ACLs):

Extended IP access list Port_1/3_E_IN
permit udp any any

Extended IP access list Port_1/3_E_OUT
permit udp any any
```

Syntax: show dot1x ip-acl

Displaying Dynamically Applied MAC Filters and IP ACLs

To display the dynamically applied MAC address filters active on an interface, enter a command such as the following:

```
BigIron# show dot1x mac-address-filter e 1/3

Port 1/3 MAC Address Filter information:
  802.1X Dynamic MAC Address Filter :
     mac filter-group 2
  Port default MAC Address Filter:
     No mac address filter is set
```

Syntax: show dot1x mac-address-filter <portnum> | all

The **all** keyword displays all dynamically applied MAC address filters active on the device.

To display the dynamically applied IP ACLs active on an interface, enter a command such as the following:

```
BigIron# show dot1x ip-acl e 1/3

Port 1/3 IP ACL information:
  802.1X dynamic IP ACL (user defined) in:
    ip access-list extended Port_1/3_E_IN in
  Port default IP ACL in:
    No inbound ip access-list is set
  802.1X dynamic IP ACL (user defined) out:
    ip access-list extended Port_1/3_E_OUT out
  Port default IP ACL out:
    No outbound ip access-list is set
```

Syntax: show dot1x ip-acl <portnum> | all

The **all** keyword displays all dynamically applied IP ACLs active on the device.

Displaying the Status of Strict Security Mode on the FastIron Edge Switch X-Series and FastIron SuperX

NOTE: This topic applies to release 02.2.00 for the FastIron Edge Switch X-Series and FastIron SuperX.

In release 02.2.00 for the FastIron Edge Switch X-Series and FastIron SuperX, the output of the **show dot1x** and **show dot1x config** commands has been enhanced to indicate whether strict security mode is enabled or disabled globally and on an interface.

Displaying the Status of Strict Security Mode Globally on the Device

To display the status of strict security mode globally on the device, enter the following command:

```
FESX448 Switch# show dot1x
PAE Capability:    Authenticator Only
system-auth-control: Enable
re-authentication: Disable
global-filter-strict-security: Enable
quiet-period:    60 Seconds
tx-period:    30 Seconds
supptimeout:    30 Seconds
servertimeout:     30 Seconds
maxreq:    2
re-authperiod:    3600 Seconds
security-hold-time: 60 Seconds
Protocol Version:    1
```

Syntax: show dot1x

Displaying the Status of Strict Security Mode on an Interface

To display the status of strict security mode on an interface, enter a command such as the following:

```
FESX448 Switch# show dot1x config e 1/3

Port 1/3 Configuration:
Authenticator PAE state:    AUTHENTICATED
Backend Authentication state:     IDLE
AdminControlledDirections:    BOTH
OperControlledDirections:    BOTH
AuthControlledPortControl:     Auto
AuthControlledPortStatus:    authorized
quiet-period:    60 Seconds
tx-period:    30 Seconds
supptimeout:    30 Seconds
servertimeout:     30 Seconds
maxreq:    2
re-authperiod:    3600 Seconds
security-hold-time: 60 Seconds
re-authentication: Disable
multiple-hosts: Disable
filter-strict-security: Enable
Protocol Version:    1
```

Syntax: show dot1x config <slotnum> <portnum>

Displaying 802.1X Multiple-Host Authentication Information (Release 07.8.00 and Later)

You can display the following information about 802.1X multiple-host authentication:

- Information about the 802.1X multiple-host configuration
- The dot1x-mac-sessions on each port
- The number of users connected on each port in a 802.1X multiple-host configuration

Displaying 802.1X Multiple-Host Configuration Information

In release 07.8.00 and later, the output of the **show dot1x** and **show dot1x config** commands displays information related to 802.1X multiple-host authentication.

The following is an example of the output of the **show dot1x** command. The information related to multiple-host authentication is highlighted in bold.

```
BigIron# show dot1x

Number of Ports enabled        : 2
Re-Authentication              : Enabled
Authentication-fail-action     : Restricted VLAN
Authentication Failure VLAN    : 111
Mac Session Aging              : Disabled for permitted MAC sessions
Mac Session max-age            : 60 seconds
Protocol Version               : 1
quiet-period                   : 5 Seconds
tx-period                      : 30 Seconds
supptimeout                    : 30 Seconds
servertimeout                  : 30 Seconds
maxreq                         : 2
re-authperiod                  : 3600 Seconds
security-hold-time             : 60 Seconds
re-authentication              : Enable
Flow based multi-user policy   : Disable
```

Syntax: show dot1x

Table 5.4 describes the bold fields in the display.

Table 5.4: Output from the show dot1x command for multiple host authentication

This Field...	Displays...
Authentication-fail-action	The configured authentication-failure action. This can be Restricted VLAN or Block Traffic.
Authentication Failure VLAN	If the authentication-failure action is Restricted VLAN, the ID of the VLAN to which unsuccessfully authenticated Client ports are assigned.
Mac Session Aging	Whether aging for dot1x-mac-sessions has been enabled or disabled for permitted or denied dot1x-mac-sessions.
Mac Session max-age	The configured software aging time for dot1x-mac-sessions.
Flow based multi-user policy	The dynamically assigned IP ACLs and MAC address filters used in the 802.1X multiple-host configuration. Note that release 07.8.00 does not support dynamically assigned IP ACLs and MAC address filters in an 802.1X multiple-host configuration. This functionality will be added in a future release.

In release 07.8.00, the output of the **show dot1x config** command for an interface was changed so it displays only the configured port control for the interface. For example:

```
BigIron# show dot1x config e 1/3

Port-Control: control-auto
```

Syntax: show dot1x config <portnum>

Table 5.5 lists the field in the display.

Table 5.5: Output from the show dot1x config command

This Field...	Displays...
Port-Control	The configured port control type for the interface. This can be one of the following: **force-authorized** – The port's controlled port is placed unconditionally in the authorized state, allowing all traffic. This is the default state for ports on the Foundry device. **force-unauthorized** – The controlled port is placed unconditionally in the unauthorized state. No authentication takes place for any connected 802.1X Clients. **auto** – The authentication status for each 802.1X Client depends on the authentication status returned from the RADIUS server.

Displaying Information About the dot1x-mac-sessions on Each Port

To display information about the dot1x-mac-sessions on each port on the device, enter the following command:

```
BigIron# show dot1x mac-session

Port            MAC/IP Address(username)        Vlan Auth    Age CAM
                                                     State       Index
-----------------------------------------------------------------------
3               0007.e90f.eb30 :kingcobra       102  permit  1   N/A
3                0007.e90f.eaa1 :cobra              102  permit   6   N/A
```

Syntax: show dot1x mac-session

Table 5.6 describes the information displayed by the **show dot1x mac-session** command.

Table 5.6: Output from the show dot1x mac-session command

This Field...	Displays...
Port	The port on which the dot1x-mac-session exists.
MAC/IP Address(username)	The MAC address of the Client and the username used for RADIUS authentication.
Vlan	The VLAN to which the port is currently assigned.

Table 5.6: Output from the show dot1x mac-session command (Continued)

This Field...	Displays...
Auth-State	The authentication state of the dot1x-mac-session. This can be one of the following: permit – The Client has been successfully authenticated, and traffic from the Client is being forwarded normally. blocked – Authentication failed for the Client, and traffic from the Client is being dropped in hardware. restricted – Authentication failed for the Client, but traffic from the Client is allowed in the restricted VLAN only. init - The Client is in is in the process of 802.1X authentication, or has not started the authentication process.
Age	The software age of the dot1x-mac-session.
CAM Index	If the MAC address is blocked, the index entry for the Layer 2 CAM entry created for this MAC address. If the MAC address is not blocked, either through successful authentication or through being placed in the restricted VLAN, then “N/A” is displayed. If the hardware aging period has expired, then “ffff” is displayed for the MAC address during the software aging period.

Displaying Information About the Ports in an 802.1X Multiple-Host Configuration

To display information about the ports in an 802.1X multiple-host configuration, enter the following command:

```
BigIron# show dot1x mac-session brief

Port            Number of  Number of        Dynamic Dynamic
                 users    Authorized users  VLAN   Filters
---------------------------------------------------------
1               0         0                 no       no
3                2          2                 yes       no
```

Syntax: show dot1x mac-session brief

The following table describes the information displayed by the **show dot1x mac-session brief** command.

Table 5.7: Output from the show dot1x mac-session brief command

This Field...	Displays...
Port	Information about the users connected to each port.
Number of users	The number of users connected to the port.
Number of Authorized users	The number of users connected to the port that have been successfully authenticated.
Dynamic VLAN	Whether the port is a member of a RADIUS-specified VLAN.
Dynamic Filters	Whether RADIUS-specified IP ACLs or MAC address filters have been applied to the port.

Enhancement to the show Commands

NOTE: This enhancement is supported in releases 03.3.00 and later for the FES, and releases 02.2.00 and later for the FESX and FWSX.

The output of the **show dot1x config** <portnum> and **show dot1x mac-session** commands has been enhanced in release 03.3.00 to display information related to 802.1X multiple-host authentication.

The following is an example of the new output of the **show dot1x config** command for an interface.

```
FES2402 Switch(config)# show dot1x config e 1

Port-Control                   : control-auto
filter strict security         : Enable
PVID State                     : Restricted (10)
Original PVID                  : 10
PVID mac total                 : 1
PVID mac authorized            : 0
num mac sessions               : 1
num mac authorized             : 0
```

Syntax: show dot1x config <portnum>

The following table lists the fields in the display.

Table 5.8: Output from the show dot1x config command

This Field...	Displays...
Port-Control	The configured port control type for the interface. This can be one of the following: **force-authorized** – The port's controlled port is placed unconditionally in the authorized state, allowing all traffic. This is the default state for ports on the Foundry device. **force-unauthorized** – The controlled port is placed unconditionally in the unauthorized state. No authentication takes place for any connected 802.1X Clients. **auto** – The authentication status for each 802.1X Client depends on the authentication status returned from the RADIUS server.
filter strict security	Whether strict security mode is enabled or disabled on the interface.
PVID State	The port's default VLAN ID (PVID) and the state of the port's PVID. The PVID state can one of the following: **Normal** – The port's PVID is not set by a RADIUS server, nor is it the restricted VLAN. **RADIUS** – The port's PVID was dynamically assigned by a RADIUS server. **RESTRICTED** – The port's PVID is the restricted VLAN.
Original PVID	The originally configured (not dynamically assigned) PVID for the port.
PVID mac total	The number of devices transmitting untagged traffic on the port's PVID.

Table 5.8: Output from the show dot1x config command (Continued)

This Field...	Displays...
PVID mac authorized	The number of devices transmitting untagged traffic on the port's PVID as a result of dynamic VLAN assignment.
num mac sessions	The number of dot1x-mac-sessions on the port.
num mac authorized	The number of authorized dot1x-mac-sessions on the port.

The output from the **show dot1x mac-session** command now shows the authenticator PAE state. For example:

```
FES2402 Switch# show dot1x mac-session

Port  MAC/(username)                  Vlan Auth    ACL   Age   PAE
                                           State               State
-------------------------------------------------------------------------------
1     0010.a498.24f7 :User            10   permit  none  S20   AUTHENTICATED
```

Syntax: show dot1x mac-session

Table 5.9 lists the new field in the display.

Table 5.9: Output from the show dot1x mac-session command

This Field...	Displays...
PAE State	The current status of the Authenticator PAE state machine. This can be INITIALIZE, DISCONNECTED, CONNECTING, AUTHENTICATING, AUTHENTICATED, ABORTING, HELD, FORCE_AUTH, or FORCE_UNAUTH. **Note:** When the Authenticator PAE state machine is in the AUTHENTICATING state, if the reAuthenticate, eapStart, eapLogoff, or authTimeout parameters are set to TRUE, it may place the Authenticator PAE state machine indefinitely in the ABORTING state. If this should happen, use the dot1x initialize command to initialize 802.1X port security on the port, or unplug the Client or hub connected to the port, then reconnect it.

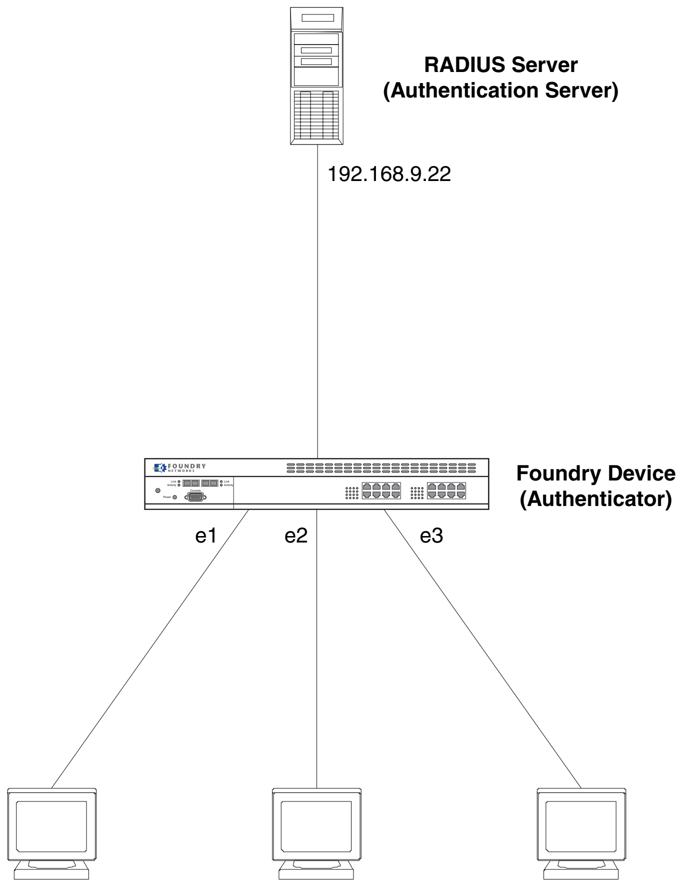
Sample 802.1X Configurations

This section illustrates a sample point-to-point configuration and a sample hub configuration that use 802.1X port security.

Point-to-Point Configuration

Figure 5.6 illustrates a sample 802.1X configuration with Clients connected to three ports on the Foundry device. In a point-to-point configuration, only one 802.1X Client can be connected to each port.

Figure 5.6 Sample point-to-point 802.1X configuration



Clients/Supplicants running 802.1X-compliant client software

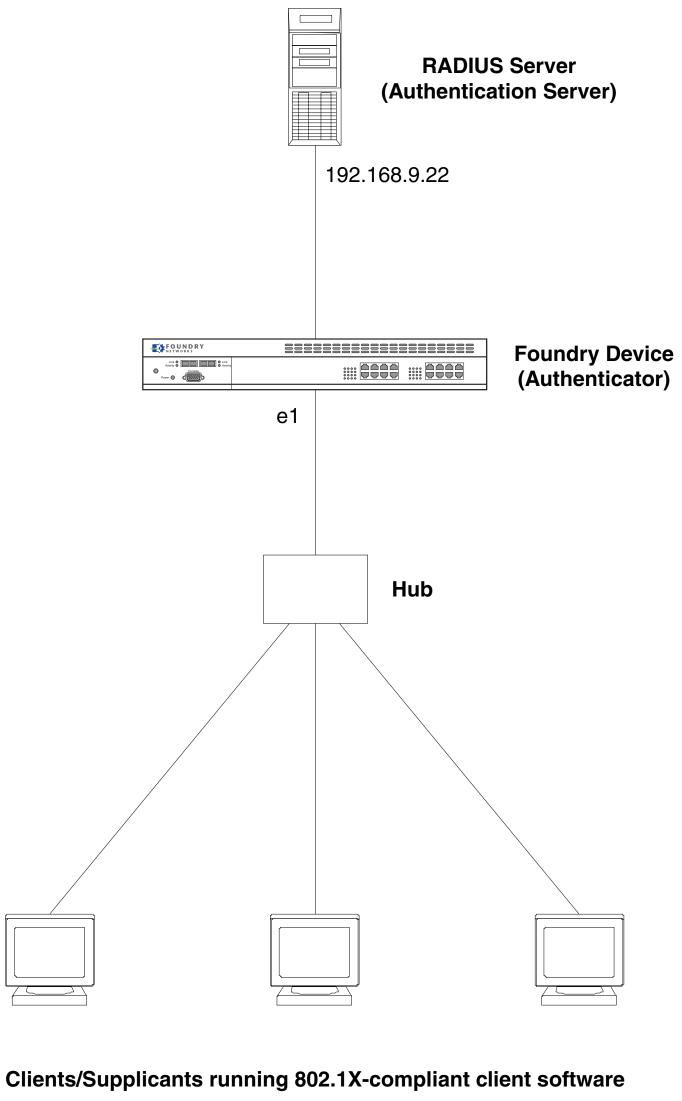
The following commands configure the Foundry device in Figure 5.6:

```
BigIron(config)# aaa authentication dot1x default radius
BigIron(config)# radius-server host 192.168.9.22 auth-port 1812 acct-port 1813
default key mirabeau dot1x

BigIron(config)# dot1x-enable e 1 to 3
BigIron(config-dot1x)# re-authentication
BigIron(config-dot1x)# timeout re-authperiod 2000
BigIron(config-dot1x)# timeout quiet-period 30
BigIron(config-dot1x)# timeout tx-period 60
BigIron(config-dot1x)# max-req 6
BigIron(config-dot1x)# exit

BigIron(config)# interface e 1
BigIron(config-if-e100-1)# dot1x port-control auto
BigIron(config-if-e100-1)# exit

BigIron(config)# interface e 2
BigIronconfig-if-e100-2)# dot1x port-control auto
BigIron(config-if-e100-2)# exit

BigIron(config)# interface e 3
BigIron(config-if-e100-3)# dot1x port-control auto
BigIron(config-if-e100-3)# exit
```

Hub Configuration (Releases Prior to 07.8.00)

Figure 5.7 illustrates a configuration where three 802.1X-enabled Clients are connected to a hub, which is connected to a port on the Foundry device. The configuration is similar to that in Figure 5.6, except that 802.1X port security is enabled on only one port, and the **multiple-hosts** command is used to allow multiple Clients on the port.

Figure 5.7 Sample 802.1X configuration using a hub



The following commands configure the Foundry device in Figure 5.7:

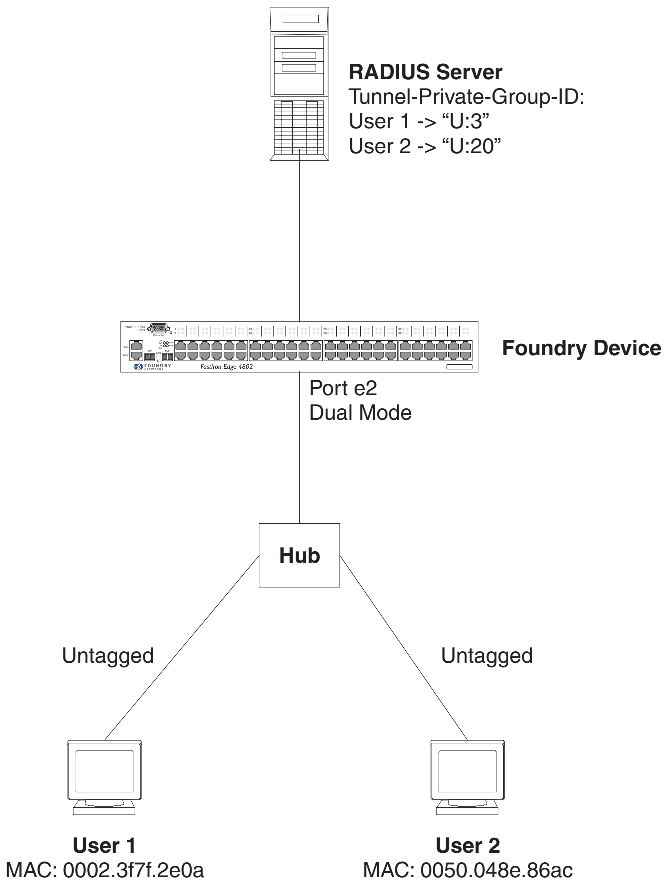
```
BigIron(config)# aaa authentication dot1x default radius
BigIron(config)# radius-server host 192.168.9.22 auth-port 1812 acct-port 1813
default key mirabeau dot1x

BigIron(config)# dot1x-enable e 1
BigIron(config-dot1x)# re-authentication
BigIron(config-dot1x)# timeout re-authperiod 2000
BigIron(config-dot1x)# timeout quiet-period 30
BigIron(config-dot1x)# timeout tx-period 60
BigIron(config-dot1x)# max-req 6
BigIron(config-dot1x)# exit
BigIron(config)# interface e 1
BigIron(config-if-e100-1)# dot1x port-control auto
BigIron(config-if-e100-1)# dot1x multiple-hosts
BigIron(config-if-e100-1)# exit
```

802.1X Authentication with Dynamic VLAN Assignment

NOTE: This configuration applies to release 03.3.00 for the FES and release 02.2.00 for the FESX and FWSX.

Figure 5.8 illustrates 802.1X authentication with dynamic VLAN assignment on an FES. In this configuration, two users' PCs are connected to a hub, which is connected to port e2 on an FES. Port e2 is configured as a dual-mode port. Both PCs transmit untagged traffic. The profile for User 1 on the RADIUS server specifies that User 1's PC should be dynamically assigned to VLAN 3. The profile for User 2 on the RADIUS server specifies that User 2's PC should be dynamically assigned to VLAN 20.

Figure 5.8 Sample configuration using 802.1X authentication with dynamic VLAN assignment



In this example, the PVID for port e2 would be changed based on the first host to be successfully authenticated. If User 1 is authenticated first, then the PVID for port e2 is changed to VLAN 3. If User 2 is authenticated first, then the PVID for port e2 is changed to VLAN 20. Since a PVID cannot be changed by RADIUS authentication after it has been dynamically assigned, if User 2 is authenticated after the port's PVID was changed to VLAN 3, then User 2 would not be able to gain access to the network.

If there were only one device connected to the port, and authentication failed for that device, it could be placed into the restricted VLAN, where it could gain access to the network.

Prior to authentication, the part of the running-config related to 802.1X authentication would be as follows:

```
dot1x-enable
 re-authentication
 servertimeout 10
 timeout re-authperiod 10
 auth-fail-action restricted-vlan
 auth-fail-vlanid 1023
 mac-session-aging no-aging permitted-mac-only
 enable ethe 2 to 4
!
!
!
interface ethernet 2
 dot1x port-control auto
 dual-mode
```

If User 1 is successfully authenticated before User 2, the **dual-mode** statement for port e2 would be changed to reflect the dynamically assigned PVID, and User 2 would not be able to gain access to the network

```
interface ethernet 2
 dot1x port-control auto
 dual-mode 3
```

Had User 2 been the first to be successfully authenticated, the VLAN ID in the **dual-mode** statement would be changed to 20, and User 1 would not be able to gain access to the network. If there were only one device connected to the port that was sending untagged traffic, and 802.1X authentication failed for that device, it would be placed in the restricted VLAN 1023, and would be able to gain access to the network. In this case the dual-mode statement in the running-config would be changed as follows:

```
interface ethernet 2
 dot1x port-control auto
 dual-mode 1023
```

Using Multi-Device Port Authentication and 802.1X Security on the Same Port

NOTE: This feature is supported in releases 02.2.00 and later for the FESX and FWSX, and Enterprise software release 08.0.00 and later.

On the FESX and FWSX running releases 02.2.00 and later, and on devices running Enterprise software releases 08.0.00 and later, you can configure the Foundry device to use multi-device port authentication and 802.1X security on the same port.

- The multi-device port authentication feature allows you to configure a Foundry device to forward or block traffic from a MAC address based on information received from a RADIUS server. Incoming traffic originating from a given MAC address is switched or forwarded by the device only if the source MAC address is successfully authenticated by a RADIUS server. The MAC address itself is used as the username and password for RADIUS authentication. A connecting user does not need to provide a specific username and password to gain access to the network.
- The IEEE 802.1X standard is a means for authenticating devices attached to LAN ports. Using 802.1X port security, you can configure a Foundry device to grant access to a port based on information supplied by a client to an authentication server.

For information on configuring the multi-device port authentication feature and 802.1X security on Foundry devices, see the July 2004 or later version of the *Foundry Security Guide*.

When both of these features are enabled on the same port, multi-device port authentication is performed prior to 802.1X authentication. If multi-device port authentication is successful, 802.1X authentication may be performed, based on the configuration of a vendor-specific attribute (VSA) in the profile for the MAC address on the RADIUS server.

When both features are configured on a port, a device connected to the port is authenticated as follows:

1. Multi-device port authentication is performed on the device to authenticate the device's MAC address.
2. If multi-device port authentication is successful for the device, then the Foundry device checks whether the RADIUS server included the Foundry-802_1x-enable VSA (described in Table 5.10) in the Access-Accept message that authenticated the device.
3. If the Foundry-802_1x-enable VSA is not present in the Access-Accept message, or is present and set to 1, then 802.1X authentication is performed for the device.
4. If the Foundry-802_1x-enable VSA is present in the Access-Accept message, and is set to 0, then 802.1X authentication is skipped. The device is authenticated, and any dynamic VLANs specified in the Access-Accept message returned during multi-device port authentication are applied to the port.
5. If 802.1X authentication is performed on the device, and is successful, then dynamic VLANs or ACLs specified in the Access-Accept message returned during 802.1X authentication are applied to the port.

If multi-device port authentication fails for a device, then by default traffic from the device is either blocked in hardware, or the device is placed in a restricted VLAN. You can optionally configure the Foundry device to perform 802.1X authentication on a device when it fails multi-device port authentication. See "Example 2" on page 5-49 for a sample configuration where this is used.

Configuring Foundry-Specific Attributes on the RADIUS Server

If the RADIUS authentication process is successful, the RADIUS server sends an Access-Accept message to the Foundry device, authenticating the device. The Access-Accept message can include Vendor-Specific Attributes (VSAs) that specify additional information about the device. If you are configuring multi-device port authentication and 802.1X authentication on the same port, then you can configure the Foundry VSAs listed in Table 5.10 on the RADIUS server.

You add these Foundry vendor-specific attributes to your RADIUS server's configuration, and configure the attributes in the individual or group profiles of the devices that will be authenticated. Foundry's Vendor-ID is 1991, with Vendor-Type 1.

Table 5.10: Foundry vendor-specific attributes for RADIUS

Attribute Name	Attribute ID	Data Type	Description
Foundry-802_1x-enable	6	integer	Specifies whether 802.1X authentication is performed when multi-device port authentication is successful for a device. This attribute can be set to one of the following: **0** Do not perform 802.1X authentication on a device that passes multi-device port authentication. Set the attribute to zero for devices that do not support 802.1X authentication. **1** Perform 802.1X authentication when a device passes multi-device port authentication. Set the attribute to one for devices that support 802.1X authentication.

Table 5.10: Foundry vendor-specific attributes for RADIUS

Attribute Name	Attribute ID	Data Type	Description
Foundry-802_1x-valid	7	integer	Specifies whether the RADIUS record is valid only for multi-device port authentication, or for both multi-device port authentication and 802.1X authentication. This attribute can be set to one of the following: **0** The RADIUS record is valid only for multi-device port authentication. Set this attribute to zero to prevent a user from using their MAC address as username and password for 802.1X authentication **1** The RADIUS record is valid for both multi-device port authentication and 802.1X authentication.

If neither of these VSAs exist in a device's profile on the RADIUS server, then by default the device is subject to multi-device port authentication (if configured), then 802.1X authentication (if configured). The RADIUS record can be used for both multi-device port authentication and 802.1X authentication.

Example Configurations

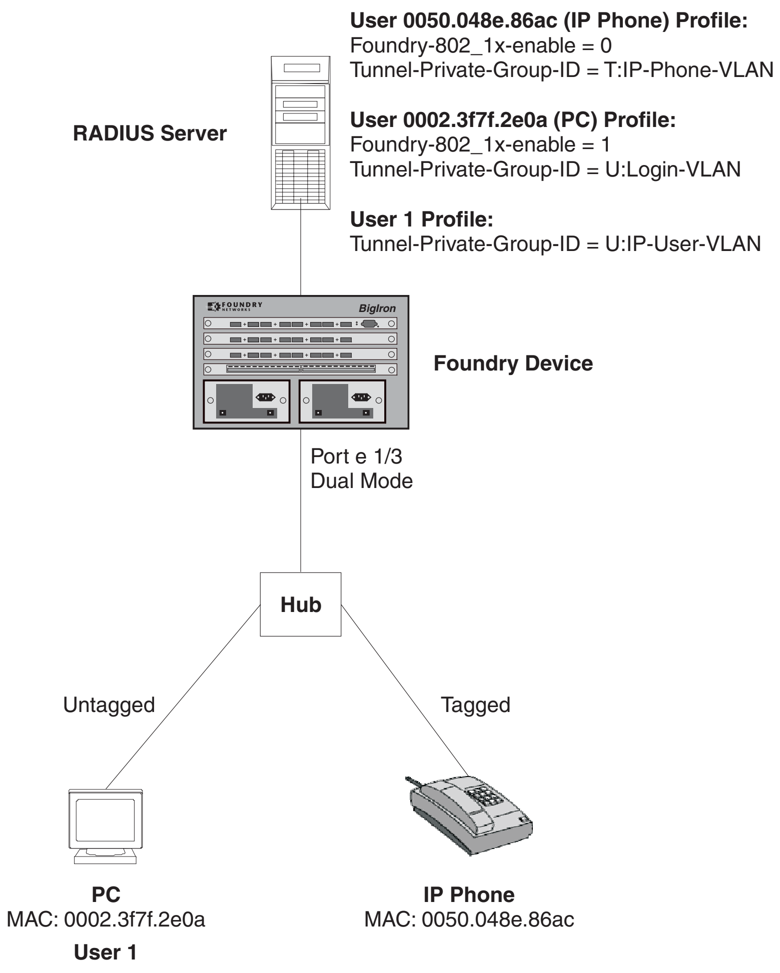
The following are two example of configurations that use multi-device port authentication and 802.1X authentication on the same port.

Example 1

Figure 5.9 illustrates a sample configuration that uses multi-device port authentication and 802.1X authentication n the same port. In this configuration, a PC and an IP phone are connected to port 1/3 on a Foundry device. Port 1/3 is configured as a dual-mode port.

The PC transmits untagged traffic, and the IP phone is configured to transmit tagged traffic (VLAN named "IP-Phone-VLAN"). The profile for the PC's MAC address on the RADIUS server specifies that the PC should be dynamically assigned to VLAN "Login-VLAN", and the profile for the IP phone specifies that it should be dynamically assigned to the VLAN named "IP-Phone-VLAN". When User 1 is successfully authenticated using 802.1X authentication, the PC is then placed in the VLAN named "User-VLAN".

Figure 5.9 Sample configuration using multi-device port authentication and 802.1X authentication on the same port


User 0050.048e.86ac (IP Phone) Profile:
Foundry-802_1x-enable = 0
Tunnel-Private-Group-ID = T:IP-Phone-VLAN
User 0002.3f7f.2e0a (PC) Profile:
Foundry-802_1x-enable = 1
Tunnel-Private-Group-ID = U:Login-VLAN
User 1 Profile:
Tunnel-Private-Group-ID = U:IP-User-VLAN
RADIUS Server
BigIron
Foundry Device
Port e 1/3
Dual Mode
Hub
Untagged
Tagged
PC
MAC: 0002.3f7f.2e0a
User 1
IP Phone
MAC: 0050.048e.86ac

When the devices attempt to connect to the network, they are first subject to multi-device port authentication.

When the IP phone's MAC address is authenticated, the Access-Accept message from the RADIUS server specifies that the IP phone's port be placed into the VLAN named "IP-Phone-VLAN". which is VLAN 7. The Foundry-802_1x-enable attribute is set to 0, meaning that 802.1X authentication is skipped for this MAC address. Port 1/3 is placed in VLAN 7 as a tagged port. No further authentication is performed.

When the PC's MAC address is authenticated, the Access-Accept message from the RADIUS server specifies that the PVID for the PC's port be changed to the VLAN named "Login-VLAN", which is VLAN 1024. The Foundry-802_1x-enable attribute is set to 1, meaning that 802.1X authentication is required for this MAC address. The PVID of the port 1/3 is temporarily changed to VLAN 1024, pending 802.1X authentication.

When User 1 attempts to connect to the network from the PC, he is subject to 802.1X authentication. If User 1 is successfully authenticated, the Access-Accept message from the RADIUS server specifies that the PVID for User 1's port be changed to the VLAN named "User-VLAN", which is VLAN 3. If 802.1X authentication for User 1 is unsuccessful, the PVID for port 1/3 is changed to that of the restricted VLAN, which is 1023, or untagged traffic from port e1 can be blocked in hardware.

Prior to authentication of the PC, the part of the running-config related to port 1/3 would be as follows:

```
interface ethernet 1/3
 dot1x port-control auto
 dual-mode
```

When the PC is authenticated using multi-device port authentication, the **dual-mode** statement for port 1/3 would be changed to reflect the dynamically assigned PVID of the “Login-VLAN”, which is VLAN 1024:

```
interface ethernet 1/3
 dot1x port-control auto
 dual-mode 1024
```

After User 1 is authenticated using 802.1X authentication, the **dual-mode** statement for port 1/3 would be changed to reflect the dynamically assigned PVID of the “User-VLAN”, which is VLAN 3:

```
interface ethernet 1/3
 dot1x port-control auto
 dual-mode 3
```

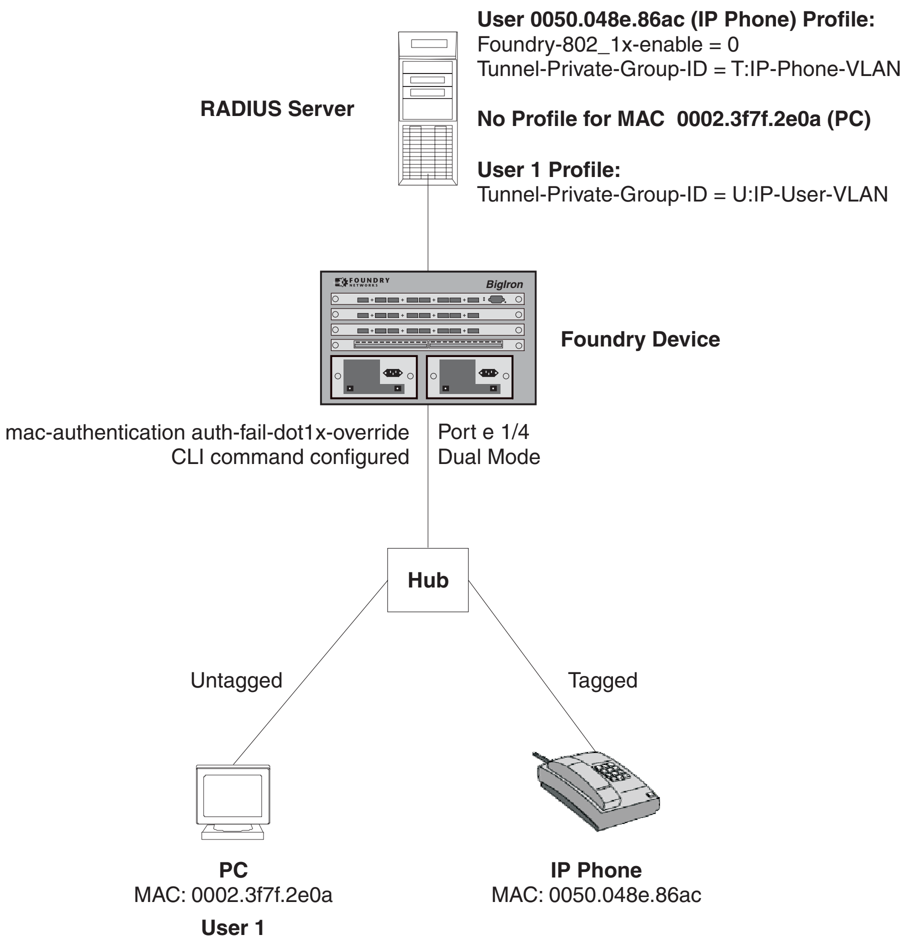
Example 2

The configuration in Figure 5.9 requires that you create a profile on the RADIUS server for each MAC address to which a device or user can connect to the network. In a large network, this can be difficult to implement and maintain.

As an alternative, you can create MAC address profiles only for those devices that do not support 802.1X authentication, such as IP phones and printers, and configure the Foundry device to perform 802.1X authentication for the other devices that do not have MAC address profiles, such as user PCs. To do this, you configure the Foundry device to perform 802.1X authentication when a device fails multi-device port authentication.

Figure 5.10 shows a configuration where multi-device port authentication is performed for an IP phone, and 802.1X authentication is performed for a user’s PC. There is a profile on the RADIUS server for the IP phone’s MAC address, but not for the PC’s MAC address.

Figure 5.10 Sample configuration where 802.1X authentication is performed when a device fails multi-device port authentication


RADIUS Server
User 0050.048e.86ac (IP Phone) Profile:
Foundry-802_1x-enable = 0
Tunnel-Private-Group-ID = T:IP-Phone-VLAN
No Profile for MAC 0002.3f7f.2e0a (PC)
User 1 Profile:
Tunnel-Private-Group-ID = U:IP-User-VLAN
Foundry Device
BigIron
mac-authentication auth-fail-dot1x-override
CLI command configured
Port e 1/4
Dual Mode
Hub
Untagged
Tagged
PC
MAC: 0002.3f7f.2e0a
User 1
IP Phone
MAC: 0050.048e.86ac

Multi-device port authentication is initially performed for both devices. The IP phone's MAC address has a profile on the RADIUS server. This profile indicates that 802.1X authentication should be skipped for this device, and that the device's port be placed into the VLAN named “IP-Phone-VLAN”.

Since there is no profile for the PC's MAC address on the RADIUS server, multi-device port authentication for this MAC address fails. Ordinarily, this would mean that the PVID for the port would be changed to that of the restricted VLAN, or untagged traffic on the port would be blocked in hardware. However, the Foundry device is configured to perform 802.1X authentication when a device fails multi-device port authentication, so when User 1 attempts to connect to the network from the PC, he is subject to 802.1X authentication. If User 1 is successfully authenticated, the PVID for port 1/4 is changed to the VLAN named “User-VLAN”.

To configure the Foundry device to perform 802.1X authentication when a device fails multi-device port authentication, enter the following command:

```
BigIron(config)# mac-authentication auth-fail-dot1x-override
```

Syntax: [no] mac-authentication auth-fail-dot1x-override

Chapter 6
Using the MAC Port Security Feature

Overview

You can configure the Foundry device to learn a limited number of “secure” MAC addresses on an interface. The interface will forward only packets with source MAC addresses that match these secure addresses. The secure MAC addresses can be specified manually, or the Foundry device can learn them automatically. After the device reaches the limit for the number of secure MAC addresses it can learn on the interface, if the interface then receives a packet with a source MAC address that is different from any of the secure learned addresses, it is considered a security violation.

When a security violation occurs, a Syslog entry and an SNMP trap are generated. In addition, the device takes one of two actions: either drops packets from the violating address (and allows packets from the secure addresses), or disables the port altogether for a specified amount of time. You specify which of these actions takes place.

The secure MAC addresses are not flushed when an interface is disabled and brought up again. The secure addresses can be kept secure permanently (the default), or can be configured to age out, at which time they are no longer secure. You can configure the device to automatically save the list of secure MAC addresses to the startup-config file at specified intervals, allowing addresses to be kept secure across system restarts.

The port security feature applies only to Ethernet interfaces.

Local and Global Resources

The port security feature uses a concept of local and global “resources” to determine how many MAC addresses can be secured on each interface. In this context, a “resource” is the ability to store one secure MAC address entry. Each interface is allocated 64 local resources. Additional global resources are shared among all the interfaces on the device.

When the port security feature is enabled, the interface can store 1 secure MAC address. You can increase the number of MAC addresses that can be secured using local resources to a maximum of 64.

Besides the maximum of 64 local resources available to an interface, there are additional global resources. Depending on flash memory size, a device can have 1024, 2048, or 4096 global resources available. When an interface has secured enough MAC addresses to reach its limit for local resources, it can secure additional MAC addresses by using global resources. Global resources are shared among all the interfaces on a first-come, first-served basis.

The maximum number of MAC addresses any single interface can secure is 64 (the maximum number of local resources available to the interface), plus the number of global resources not allocated to other interfaces.

MAC Port Security Differences

The following summarizes the differences in the MAC port security implementation between the FastIron SuperX running release 02.2.00 and the BigIron devices:

- The MAC port security feature is not supported for ports that are static trunk group members or ports that are configured for link aggregation.
- The FastIron SuperX does not support the **reserved-vlan-id <num>** command, which changes the default VLAN ID for the MAC port security feature.
- The following new port security commands have been added on the FastIron SuperX:
    - The command **show port security ethernet <slot_num> <port_num> restricted-macs** displays a list of restricted MAC addresses on the port.
    - The command **clear port security restricted-macs [all | ethernet <port_num>]** clears all restricted MAC addresses from the port.
    - The command **clear port security statistics [all | ethernet <slot_num> <port_num>]** clears violation statistics for the port.
- You can specify a number of minutes that the device drops packets from a violating address. To do this, use the **violation restrict <age>** command. The <age> can be from 0 – 1440 minutes. The default is 5 minutes. Specifying 0 drops packets from the violating address permanently.

  Aging for restricted MAC addresses is done in software. There can be a worst case inaccuracy of one minute from the specified time.

  The restricted MAC addresses are denied in hardware in the SuperX.
- When the **restrict** option is used, the maximum number of MAC addresses that can be restricted is 128. If the number of violating MAC addresses exceeds this number, the port is shut down. An SNMP trap and the following Syslog message are generated: "Port Security violation restrict limit 128 exceeded on interface ethernet <port_id>". This is followed by a port shutdown Syslog message and trap.
- The SNMP trap generated for restricted MAC addresses has been enhanced to indicate the VLAN ID associated with the MAC address, as well as the port number and MAC address.
- When specifying a secure MAC address on a tagged port, you must specify the VLAN ID as well. To do this, use the command **secure-mac-address <mac-address> <vlan-id>**.

NOTE: If MAC port security is enabled on a port, and you dynamically change the VLAN membership of the port, make sure that you also change the VLAN ID specified in the **secure-mac-address** configuration statement for the port.

In generation of the configuration, the **vlan-id** is generated for both tagged and untagged ports. When you display the configuration, you will see an entry for the secure MAC addresses `secure-mac-address <address> <vlan>.` For example, you may see the following line:

```
secure-mac-address 0000.1111.2222 10
```

This line means that MAC address 0000.1111.2222 on VLAN 10 is a secure MAC address.

Configuring the MAC Port Security Feature

To configure the MAC port security feature, you perform the following tasks:

- Enable the MAC port security feature
- Set the maximum number of secure MAC addresses for an interface
- Set the port security age timer
- Specify secure MAC addresses

- Configure the device to automatically save secure MAC addresses to the startup-config file
- Specify the action taken when a security violation occurs
- Deny specific MAC addresses

Enabling the MAC Port Security Feature

By default, the MAC port security feature is disabled on all interfaces. You can enable or disable the feature globally on all interfaces at once or on individual interfaces.

To enable the feature on all interfaces at once:

```
BigIron(config)# port security
BigIron(config-port-security)# enable
```

To disable the feature on all interfaces at once:

```
BigIron(config)# port security
BigIron(config-port-security)# no enable
```

To enable the feature on a specific interface:

```
BigIron(config)# int e 7/11
BigIron(config-if-e100-7/11)# port security
BigIron(config-port-security-e100-7/11)# enable
```

Syntax: port security

Syntax: [no] enable

Setting the Maximum Number of Secure MAC Addresses for an Interface

When the port security feature is enabled, the interface can store 1 secure MAC address. You can increase the number of MAC addresses that can be secured to a maximum of 64, plus the total number of global resources available.

For example, to configure interface 7/11 to have a maximum of 10 secure MAC addresses:

```
BigIron(config)# int e 7/11
BigIron(config-if-e100-7/11)# port security
BigIron(config-if-e100-7/11)# maximum 10
```

Syntax: maximum <number-of-addresses>

The <number-of-addresses> parameter can be set to a number from 0 – (64 + the total number of global resources available) The total number of global resources is 2048 or 4096, depending on flash memory size. Setting the parameter to 0 prevents any addresses from being learned. The default is 1.

Setting the Port Security Age Timer

By default, the learned MAC addresses stay secure indefinitely. You can optionally configure the device to age out secure MAC addresses after a specified amount of time.

To set the port security age timer to 10 minutes on all interfaces:

```
BigIron(config)# port security
BigIron(config-port-security)# age 10
```

To set the port security age timer to 10 minutes on a specific interface:

```
BigIron(config)# int e 7/11
BigIron(config-if-e100-7/11)# port security
BigIron(config-port-security-e100-7/11)# age 10
```

Syntax: [no] age <minutes>

The default is 0 (never age out secure MAC addresses).

Specifying Secure MAC Addresses

To specify a secure MAC address on an interface, enter commands such as the following:

```
BigIron(config)# int e 7/11
BigIron(config-if-e100-7/11)# port security
BigIron(config-port-security-e100-7/11)# secure 0050.DA18.747C
```

Syntax: [no] secure <mac-address>

Autosaving Secure MAC Addresses to the Startup-Config File

The learned MAC addresses can automatically be saved to the startup-config file at specified intervals. For example, to automatically save learned secure MAC addresses on the device every twenty minutes, enter the following commands:

```
BigIron(config)# port security
BigIron(config-port-security)# autosave 20
```

Syntax: [no] autosave <minutes>

You can specify from 15 – 1440 minutes. By default, secure MAC addresses are not autosaved to the startup-config file.

Specifying the Action Taken when a Security Violation Occurs

A security violation can occur when a user tries to plug into a port where a MAC address is already locked, or the maximum number of secure MAC addresses has been exceeded. When a security violation occurs, an SNMP trap and Syslog message are generated.

In addition, you configure the device to take one of two actions when a security violation occurs: either drop packets from the violating address (and allow packets from secure addresses), or disable the port altogether for a specified amount of time.

To configure the device to drop packets from a violating address and allow packets from secure addresses:

```
BigIron(config)# int e 7/11
BigIron(config-if-e100-7/11)# port security
BigIron(config-port-security-e100-7/11)# violation restrict
```

Syntax: violation restrict

To shut down the port for 5 minutes when a security violation occurs:

```
BigIron(config)# int e 7/11
BigIron(config-if-e100-7/11)# port security
BigIron(config-port-security-e100-7/11)# violation shutdown 5
```

Syntax: violation shutdown <minutes>

You can specify from 0 – 1440 minutes. Specifying 0 shuts down the port permanently when a security violation occurs.

NOTE: When using this feature with a 24-port 10/100 module (part number B24E) only the **shutdown** option is supported. The **restrict** option is not supported on the B24E.

Denying Specific MAC Addresses

On BigIron MG8 and NetIron 40G devices running software release 02.2.01 and later, you can configure the *violation deny mode*. The violation deny mode allows you to deny MAC addresses on a global level or on a per port level.

Denying MAC Addresses Globally

To deny a specific MAC address globally, enable the violation deny mode, then spcify the MAC address to be denied.

```
BigIron(config)# port security
BigIron(config-port-security)# violation deny
BigIron(config-port-security)# deny-mac-address 0000.0000.0001 2
```

Global denied secure MAC addresses are denied system-wide. These MAC entries are added to the MAC table as deny entries, when a flow is received and are the only MAC addresses that are denied. All other MAC addresses are allowed.

A maximum of 512 deny MAC addresses can be configured on a global level.

Denying MAC Addresses on an Interface

You can specify which MAC addresses can be denied on an interface.

```
BigIron(config)# int e 7/11
BigIron(config-if-e100-7/11)# port security
BigIron(config-port-security-e100-7/11)# violation deny
BigIron(config-port-security-e100-7/11)# deny-mac-addr 0000.1111.2222 4
```

Only the configured MAC addresses are denied on the specified interface. All other MAC addresses are allowed.

A maximum of 64 deny MAC addresses can be configured at an interface level.

Displaying MAC Addresses that Have Been Denied

Use the **show port security global-deny** command to display all the MAC addresses that have been denied globally. Use the **show port security denied mac** command to display all the denied MAC addresses

Displaying Port Security Information

You can display the following information about the port security feature:

- The secure MAC addresses that have been saved to the startup-config file by the autosave feature
- The port security settings for an individual port or for all the ports on a specified module
- The secure MAC addresses configured on the device
- Port security statistics for an interface or for a module

Displaying Autosaved MAC Addresses

To display the secure MAC addresses that have been saved to the configuration by the autosave feature, enter the following command:

```
BigIron# show port security autosave
```

Syntax: show port security autosave

Displaying Port Security Settings

You can display the port security settings for an individual port or for all the ports on a specified module. For example, to display the port security settings for port 7/11, enter the following command:

```
BigIron# show port security e 7/11
Port  Security Violation Shutdown-Time Age-Time  Max-MAC
----- -------- --------- ------------- --------- -------
 7/11 disabled  shutdown            10        10       1
```

Syntax: show port security <module> | <portnum>

This command displays the following information

Table 6.1: Output from the show port security <module> command

This Field...	Displays...
Port	The slot and port number of the interface.
Security	Whether the port security feature has been enabled on the interface.
Violation	The action to be undertaken when a security violation occurs, either “shutdown” or “restrict”.
Shutdown-Time	The number of seconds a port is shut down following a security violation, if the port is set to “shutdown” when a violation occurs.
Age-Time	The amount of time, in minutes, MAC addresses learned on the port will remain secure.
Max-MAC	The maximum number of secure MAC addresses that can be learned on the interface.

Displaying the Secure MAC Addresses on the Device

To list the secure MAC addresses configured on the device, enter the following command:

```
BigIron(config)# show port security mac
Port  Num-Addr Secure-Src-Addr Resource Age-Left  Shutdown/Time-Left
----- -------- --------------- -------- --------- ------------------
 7/11        1  0050.da18.747c    Local        10       no
```

Syntax: show port security mac

This command displays the following information:

Table 6.2: Output from the show port security mac command

This Field...	Displays...
Port	The slot and port number of the interface.
Num-Addr	The number of MAC addresses secured on this interface.
Secure-Src-Addr	The secure MAC address.
Resource	Whether the address was secured using a local or global resource. See “Local and Global Resources” on page 6-1 for more information.
Age-Left	The number of minutes the MAC address will remain secure.
Shutdown/Time-Left	Whether the interface has been shut down due to a security violation and the number of seconds before it is enabled again.

Displaying Port Security Statistics

You can display port security statistics for an interface or for a module.

For example, to display port security statistics for interface 7/11:

```
BigIron# show port security statistics e 7/11
Port  Total-Addrs Maximum-Addrs Violation Shutdown/Time-Left
----- ----------- ------------- --------- ------------------
 7/11           1             1         0       no
```

Syntax: show port security statistics <portnum>

Table 6.3: Output from the show port security statistics <portnum> command

This Field...	Displays...
Port	The slot and port number of the interface.
Total-Addrs	The total number of secure MAC addresses on the interface.
Maximum-Addrs	The maximum number of secure MAC addresses on the interface.
Violation	The number of security violations on the port.
Shutdown/Time-Left	Whether the port has been shut down due to a security violation and the number of seconds before it is enabled again.

To display port security statistics for a module, enter the following command:

```
BigIron# show port security statistics 7
Module 7:
  Total ports: 0
  Total MAC address(es): 0
  Total violations: 0
  Total shutdown ports 0
```

Syntax: show port security statistics <module>

Table 6.4: Output from the show port security statistics <module> command

This Field...	Displays...
Total ports:	The number of ports on the module.
Total MAC address(es):	The total number of secure MAC addresses on the module.
Total violations:	The number of security violations encountered on the module.
Total shutdown ports:	The number of ports on the module shut down as a result of security violations.

Chapter 7
Configuring Multi-Device Port Authentication

Multi-device port authentication is a way to configure a Foundry device to forward or block traffic from a MAC address based on information received from a RADIUS server.

This chapter is divided into the following sections:

- "How Multi-Device Port Authentication Works" below explains basic concepts about multi-device port authentication.
- "Using Multi-Device Port Authentication and 802.1X Security on the Same Port" on page 7-5
- "Configuring Multi-Device Port Authentication" on page 7-6 describes how to set up multi-device port authentication on Foundry devices using the Command Line Interface (CLI).
- "Displaying Multi-Device Port Authentication Information" on page 7-12 describes the commands used to display information about a multi-device port authentication configuration.
- "Sample Configurations" on page 7-18.

NOTE: This feature is supported on the following Foundry devices:

- Devices running Enterprise software release 07.6.06 and higher
- FES devices running software release 03.3.00 and higher
- FESX and FWSX devices running software release 02.2.00 and higher
- FSX devices running software release 02.3.01 and higher
- BigIron MG8 and NetIron 40G running software release 02.2.01 and higher; however, options and features that are specific to the FastIron Edge Switches are not supported on the BigIron MG8 and NetIron 40G.

How Multi-Device Port Authentication Works

The multi-device port authentication feature is a mechanism by which incoming traffic originating from a specific MAC address is switched or forwarded by the device only if the source MAC address is successfully authenticated by a RADIUS server. The MAC address itself is used as the username and password for RADIUS authentication; the user does not need to provide a specific username and password to gain access to the network. If RADIUS authentication for the MAC address is successful, traffic from the MAC address is forwarded in hardware.

If the RADIUS server cannot validate the user's MAC address, then it is considered an authentication failure, and a specified authentication-failure action can be taken. The default authentication-failure action is to drop traffic from the non-authenticated MAC address in hardware. You can also configure the device to move the port on which the non-authenticated MAC address was learned into a restricted or “guest” VLAN, which may have limited access to the network.

RADIUS Authentication

The multi-device port authentication feature communicates with the RADIUS server to authenticate a newly found MAC address. The Foundry device supports multiple RADIUS servers; if communication with one of the RADIUS servers times out, the others are tried in sequential order. If a response from a RADIUS server is not received within a specified time (by default, 3 seconds) the RADIUS session times out, and the device retries the request up to three times. If no response is received, the next RADIUS server is chosen, and the request is sent for authentication.

The RADIUS server is configured with the usernames and passwords of authenticated users. For multi-device port authentication, the username and password is the MAC address itself; that is, the device uses the MAC address for both the username and the password in the request sent to the RADIUS server. For example, given a MAC address of 0007e90feaa1, the users file on the RADIUS server would be configured with a username and password both set to 0007e90feaa1. When traffic from this MAC address is encountered on a MAC-authentication-enabled interface, the device sends the RADIUS server an Access-Request message with 0007e90feaa1 as both the username and password. The format of the MAC address sent to the RADIUS server is configurable through the CLI.

The request for authentication from the RADIUS server is successful only if the username and password provided in the request matches an entry in the users database on the RADIUS server. When this happens, the RADIUS server returns an Access-Accept message back to the Foundry device. When the RADIUS server returns an Access-Accept message for a MAC address, that MAC address is considered authenticated, and traffic from the MAC address is forwarded normally by the Foundry device.

Authentication-Failure Actions

If the MAC address does not match the username and password of an entry in the users database on the RADIUS server, then the RADIUS server returns an Access-Reject message. When this happens, it is considered an authentication failure for the MAC address. When an authentication failure occurs, the Foundry device can either drop traffic from the MAC address in hardware (the default), or move the port on which the traffic was received to a restricted VLAN.

Supported RADIUS Attributes

The IronCore, JetCore, FastIron Edge Switch, and FastIron Edge Switch X-Series devices support the following RADIUS attributes for multice-device port authetnication:

- Username (1) – RFC 2865
- NAS-IP-Address (4) – RFC 2865
- NAS-Port (5) – RFC 2865
- Service-Type (6) – RFC 2865
- FilterId (11) – RFC 2865
- Framed-MTU (12) – RFC 2865
- State (24) – RFC 2865
- Vendor-Specific (26) – RFC 2865
- Session-Timeout (27) – RFC 2865
- Termination-Action (29) – RFC 2865
- Calling-Station-ID (31) – RFC 2865
- NAS-Port-Type (61) š RFC 2865
- Tunnel-Type (64) – RFC 2868
- Tunnel-Medium-Type (65) – RFC 2868

- EAP Message (79) – RFC 2579
- Message-Authenticator (80) RFC 3579
- Tunnel-Private-Group-Id (81) – RFC 2868
- NAS-Port-id (87) – RFC2869

Dynamic VLAN Assignment

The multi-device port authentication feature supports ***dynamic VLAN assignment***, where a port can be placed in a VLAN based on the MAC address learned on that interface. When a MAC address is successfully authenticated, the RADIUS server sends the Foundry device a RADIUS Access-Accept message that allows the Foundry device to forward traffic from that MAC address. The RADIUS Access-Accept message can also contain attributes set for the MAC address in its access profile on the RADIUS server.

If one of the attributes in the Access-Accept message specifies a VLAN identifier, and this VLAN is available on the Foundry device, the port is moved from its default VLAN to the specified VLAN.

To enable dynamic VLAN assignment for authenticated MAC addresses, you must add the following attributes to the profile for the MAC address on the RADIUS server, then enable dynamic VLAN assignment on multi-device port authentication-enabled interfaces.

Attribute Name	**Type**	**Value**
Tunnel-Type	064	13 (decimal) – VLAN
Tunnel-Medium-Type	065	6 (decimal) – 802
Tunnel-Private-Group-ID	081	<vlan-name> (string) – either the name or the number of a VLAN configured on the Foundry device.

Support for Authenticating Multiple MAC Addresses on an Interface

The multi-device port authentication feature allows multiple MAC addresses to be authenticated or denied authentication on each interface. The maximum number of MAC addresses that can be authenticated on each interface is limited only by the amount of system resources available on the Foundry device.

Additions to Multi-Device Port Authentication

NOTE: The topics under this section apply only to releases 03.3.00 and later for the FES, releases 02.2.00 and later for the FESX and FWSX, and releases 02.3.01 and later for the FSX.

In the above mentioned platforms and their associated releases , multi-device port authentication works as documented in this section, with the following additions:

- Configurable hardware aging period for blocked MAC addresses
- Dynamically assigning a port to multiple VLANs

Configurable Hardware Aging Period for Blocked MAC Addresses

When the Foundry device is configured to drop traffic from non-authenticated MAC addresses, traffic from the blocked MAC addresses is dropped in hardware, without being sent to the CPU. A Layer 2 hardware entry is created that drops traffic from the MAC address in hardware. If no traffic is received from the MAC address for a certain amount of time, this Layer 2 hardware entry is aged out. If traffic is subsequently received from the MAC address, then an attempt can be made to authenticate the MAC address again.

Aging of the Layer 2 hardware entry for a blocked MAC address occurs in two phases, known as hardware aging and software aging.

On BigIron/FastIron devices, the hardware aging period for blocked MAC addresses is fixed at 70 seconds and is non-configurable. (The hardware aging time for non-blocked MAC addresses is the length of time specified with the **mac-age** command.) The software aging period for blocked MAC addresses is configurable through the CLI, with the **mac-authentication max-age** command. Once the hardware aging period ends, the software aging period begins. When the software aging period ends, the blocked MAC address ages out, and can be authenticated again if the Foundry device receives traffic from the MAC address.

On FES, FESX, FSX, and FWSX devices, the hardware aging period for blocked MAC addresses is not fixed at 70 seconds. The hardware aging period for blocked MAC addresses is equal to the length of time specified with the **mac-age** command. As on BigIron/FastIron devices, once the hardware aging period ends, the software aging period begins. When the software aging period ends, the blocked MAC address ages out, and can be authenticated again if the device receives traffic from the MAC address.

To change the hardware aging period for blocked MAC addresses, enter a command such as the following:

```
FESX424 router(config)# mac-authentication hw-deny-age 10
```

Syntax: [no] mac-authentication hw-deny-age <num>

The <num> parameter is a value from 1 to 65535 seconds. The default is 70 seconds.

Dynamically Assigning a Port to Multiple VLANs

On Foundry devices, the multi-device port authentication feature supports dynamic VLAN assignment, where a port can be placed in a VLAN based on the MAC address learned on that interface.

When a MAC address is successfully authenticated, the RADIUS server sends the Foundry device a RADIUS Access-Accept message that allows the Foundry device to forward traffic from that MAC address. The RADIUS Access-Accept message can also contain attributes set for the MAC address in its access profile on the RADIUS server. If one of the attributes in the Access-Accept message specifies a VLAN identifier, and this VLAN is available on the Foundry device, the port is moved from its default VLAN to the specified VLAN.

The Foundry device supports dynamically assigning a port to multiple VLANs, based on the results from the RADIUS server. If the RADIUS Access-Accept message specifies multiple VLAN identifiers, the FES can assign the port to the specified VLANs.

To specify VLAN identifiers on the RADIUS server, you add the following attributes to the profile for the MAC address on the RADIUS server:

Attribute Name	Type	Value
Tunnel-Type	064	13 (decimal) – VLAN
Tunnel-Medium-Type	065	6 (decimal) – 802
Tunnel-Private-Group-ID	081	<vlan-name> (string) – see below.

On BigIron/FastIron devices, the <vlan-name> value for the Tunnel-Private-Group-ID attribute can specify either the name or the number of a single VLAN configured on the Foundry device.

On the FES, FESX, FSX, and FWSX devices, the <vlan-name> value for the Tunnel-Private-Group-ID attribute can specify the name or number of one or more VLANs configured on the Foundry device.

For information about the attributes, see “Dynamic Multiple VLAN Assignment for 802.1X Ports on the FES, FESX, FSX, and FWSX” on page 5-13.

Also, see sample configuration of “Multi-Device Port Authentication with Dynamic VLAN Assignment” on page 7-18.

Using Multi-Device Port Authentication and 802.1X Security on the Same Port

On some Foundry devices, multi-device port authentication and 802.1X security can be configured on the same port.

On the BigIron MG8 and NetIron 40G

On the BigIron MG8 and NetIron 40G, multi-device port authentication and 802.1x port security can be enabled on the same interface. However, only one of these features will be used to authenticate a MAC addresses and 802.1x client. If an 802.1x client responds, the software assumes that the MAC address should be authenticated using 802.1x protocols. Multi-device port authentication for that MAC address is aborted.

On the FES

You can configure the FES to use multi-device port authentication and 802.1X security on the same port. When both of these features are enabled on the same port, multi-device port authentication is performed prior to 802.1X authentication. If multi-device port authentication is successful, 802.1X authentication may be performed, based on the configuration of a vendor-specific attribute (VSA) in the profile for the MAC address on the RADIUS server.

When both features are configured on a port, a device connected to the port is authenticated as follows:

1. Multi-device port authentication is performed on the device to authenticate the device's MAC address.
2. If multi-device port authentication is successful for the device, then the FES checks whether the RADIUS server included the Foundry-802_1x-enable VSA (described in Table 7.1) in the Access-Accept message that authenticated the device.
3. If the Foundry-802_1x-enable VSA is not present in the Access-Accept message, or is present and set to 1, then 802.1X authentication is performed for the device.
4. If the Foundry-802_1x-enable VSA is present in the Access-Accept message, and is set to 0, then 802.1X authentication is skipped. The device is authenticated, and any dynamic VLANs specified in the Access-Accept message returned during multi-device port authentication are applied to the port.
5. If 802.1X authentication is performed on the device, and is successful, then dynamic VLANs or ACLs specified in the Access-Accept message returned during 802.1X authentication are applied to the port.

If multi-device port authentication fails for a device, then by default traffic from the device is either blocked in hardware, or the device is placed in a restricted VLAN. You can optionally configure the FES to perform 802.1X authentication on a device when it fails multi-device port authentication. See "Example 2" on page 7-22 for a sample configuration where this is used.

Configuring Foundry-Specific Attributes on the RADIUS Server

If the RADIUS authentication process is successful, the RADIUS server sends an Access-Accept message to the Foundry device, authenticating the device. The Access-Accept message can include Vendor-Specific Attributes (VSAs) that specify additional information about the device. If you are configuring multi-device port authentication and 802.1X authentication on the same port, then you can configure the Foundry VSAs listed in Table 7.1 on the RADIUS server.

You add these Foundry vendor-specific attributes to your RADIUS server's configuration, and configure the attributes in the individual or group profiles of the devices that will be authenticated. Foundry's Vendor-ID is 1991, with Vendor-Type 1.

Table 7.1: Foundry vendor-specific attributes for RADIUS

Attribute Name	Attribute ID	Data Type	Description
Foundry-802_1x-enable	6	integer	Specifies whether 802.1X authentication is performed when multi-device port authentication is successful for a device. This attribute can be set to one of the following: **0** Do not perform 802.1X authentication on a device that passes multi-device port authentication. Set the attribute to zero for devices that do not support 802.1X authentication. **1** Perform 802.1X authentication when a device passes multi-device port authentication. Set the attribute to one for devices that support 802.1X authentication.
Foundry-802_1x-valid	7	integer	Specifies whether the RADIUS record is valid only for multi-device port authentication, or for both multi-device port authentication and 802.1X authentication. This attribute can be set to one of the following: **0** The RADIUS record is valid only for multi-device port authentication. Set this attribute to zero to prevent a user from using their MAC address as username and password for 802.1X authentication **1** The RADIUS record is valid for both multi-device port authentication and 802.1X authentication.

If neither of these VSAs exist in a device's profile on the RADIUS server, then by default the device is subject to multi-device port authentication (if configured), then 802.1X authentication (if configured). The RADIUS record can be used for both multi-device port authentication and 802.1X authentication.

See sample configuration of "Examples of Multi-Device Port Authentication and 802.1X Authentication Configuration on the Same Port (FES)" on page 7-20.

Configuring Multi-Device Port Authentication

Configuring multi-device port authentication on the Foundry device consists of the following tasks:

- Enabling multi-device port authentication globally and on individual interfaces
- Specifying the format of the MAC addresses sent to the RADIUS server (optional)
- Specifying the authentication-failure action (optional)
- Defining MAC address filters (optional)
- Configuring dynamic VLAN assignment (optional)

- Specifying to which VLAN a port is moved after its RADIUS-specified VLAN assignment expires (optional)
- Saving dynamic VLAN assignments to the running-config file (optional)
- Enabling denial of service attack protection (optional)
- Clearing authenticated MAC addresses (optional)
- Disabling aging for authenticated MAC addresses (optional)
- Specifying the aging time for blocked MAC addresses (optional)

Enabling Multi-Device Port Authentication

To enable multi-device port authentication, you first enable the feature globally on the device. On some Foundry devices, you can then enable the feature on individual interfaces.

Globally Enabling Multi-Device Port Authentication

To globally enable multi-device port authentication on the device, enter the following command:

```
BigIron(config)# mac-authentication enable
```

Syntax: [no] mac-authentication enable

Enabling Multi-Device Port Authentication on Interfaces

NOTE: The following commands are not available on the BigIron MG8 and NetIron 40G.

To enable multi-device port authentication on an individual interface, enter a command such as the following:

```
BigIron(config)# mac-authentication enable ethernet 3/1
```

Syntax: [no] mac-authentication enable <portnum> | all

The **all** option enables the feature on all interfaces at once.

You can enable the feature on an interface at the interface CONFIG level. For example:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication enable
```

Syntax: [no] mac-authentication enable

You can also configure multi-device port authentication commands on a range of interfaces. For example:

```
BigIron(config)# int e 3/1 to 3/12
BigIron(config-mif-3/1-3/12)# mac-authentication enable
```

Specifying the Format of the MAC Addresses Sent to the RADIUS Server

When multi-device port authentication is configured, the Foundry device authenticates MAC addresses by sending username and password information to a RADIUS server. The username and password is the MAC address itself; that is, the device uses the MAC address for both the username and the password in the request sent to the RADIUS server.

By default, the MAC address is sent to the RADIUS server in the format *xxxxxxxxxxxx*. You can optionally configure the device to send the MAC address to the RADIUS server in the format *xx-xx-xx-xx-xx-xx*, or the format *xxxx.xxxx.xxxx*. To do this, enter a command such as the following:

```
BigIron(config)# mac-authentication auth-passwd-format xxxx.xxxx.xxxx
```

Syntax: [no] mac-authentication auth-passwd-format xxxx.xxxx.xxxx | xx-xx-xx-xx-xx-xx | xxxxxxxxxxxx

Specifying the Authentication-Failure Action

When RADIUS authentication for a MAC address fails, you can configure the device to perform one of two actions:

- Drop traffic from the MAC address in hardware (the default)

- Move the port on which the traffic was received to a restricted VLAN

To configure the device to move the port to a restricted VLAN when multi-device port authentication fails, enter commands such as the following:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication auth-fail-action restrict-vlan 100
```

Syntax: [no] mac-authentication auth-fail-action restrict-vlan [<vlan-id>]

If the ID for the restricted VLAN is not specified at the interface level, the global restricted VLAN ID applies for the interface.

To specify the VLAN ID of the restricted VLAN globally, enter the following command:

```
BigIron(config)# mac-authentication auth-fail-vlan-id 200
```

Syntax: [no] mac-authentication auth-fail-vlan-id <vlan-id>

The command above applies globally to all MAC-authentication-enabled interfaces.

Note that the restricted VLAN must already exist on the device. You cannot configure the restricted VLAN to be a non-existent VLAN. If the port is a tagged or dual-mode port, you cannot use a restricted VLAN as the authentication-failure action.

To configure the device to drop traffic from non-authenticated MAC addresses in hardware, enter commands such as the following:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication auth-fail-action block-traffic
```

Syntax: [no] mac-authentication auth-fail-action block-traffic

Dropping traffic from non-authenticated MAC addresses is the default behavior when multi-device port authentication is enabled.

Defining MAC Address Filters

You can specify MAC addresses that do not have to go through multi-device port authentication. These MAC addresses are considered pre-authenticated, and are not subject to RADIUS authentication. To do this, you can define MAC address filters that specify the MAC addresses to exclude from multi-device port authentication.

You should use a MAC address filter when the RADIUS server itself is connected to an interface where multi-device port authentication is enabled. If a MAC address filter is not defined for the MAC address of the RADIUS server and applied on the interface, the RADIUS authentication process would fail since the device would drop all packets from the RADIUS server itself.

For example, the following command defines a MAC address filter for address 0010.dc58.aca4:

```
BigIron(config)# mac-authentication mac-filter 1 permit 0010.dc58.aca4
```

Syntax: [no] mac-authentication mac-filter <filter>

The following commands apply the MAC address filter on an interface so that address 0010.dc58.aca4 is excluded from multi-device port authentication:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication apply-mac-auth-filter 1
```

Syntax: [no] mac-authentication apply-mac-auth-filter <filter-id>

Configuring Dynamic VLAN Assignment

An interface can be dynamically assigned to a VLAN based on the MAC address learned on that interface. When a MAC address is successfully authenticated, the RADIUS server sends the Foundry device a RADIUS Access-Accept message that allows the Foundry device to forward traffic from that MAC address. The RADIUS Access-Accept message can also contain attributes set for the MAC address in its access profile on the RADIUS server.

If one of the attributes in the Access-Accept message specifies a VLAN identifier, and this VLAN is available on the Foundry device, the port is moved from its default VLAN to the specified VLAN.

To enable dynamic VLAN assignment for authenticated MAC addresses, you must add the following attributes to the profile for the MAC address on the RADIUS server, then enable dynamic VLAN assignment on multi-device port authentication-enabled interfaces. See "Dynamic VLAN Assignment" on page 7-3 for a list of the attributes that must be set on the RADIUS server

To enable dynamic VLAN assignment on a multi-device port authentication-enabled interface, enter commands such as the following:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication enable-dynamic-vlan
```

Syntax: [no] mac-authentication enable-dynamic-vlan

If a previous authentication attempt for a MAC address failed, and as a result the port was placed in the restricted VLAN, but a subsequent authentication attempt was successful, the RADIUS Access-Accept message may specify a VLAN for the port. By default, the Foundry device moves the port out of the restricted VLAN and into the RADIUS-specified VLAN. You can optionally configure the device to ignore the RADIUS-specified VLAN in the RADIUS Access-Accept message, and leave the port in the restricted VLAN.

To do this, enter the following command:

```
BigIron(config)# mac-authentication no-override-restrict-vlan
```

Syntax: [no] mac-authentication no-override-restrict-vlan

Notes:

- For untagged ports, if the VLAN ID provided by the RADIUS server is valid, then the port is removed from its current VLAN and moved to the RADIUS-specified VLAN as an untagged port.
- For tagged ports, if the VLAN ID provided by the RADIUS server is valid, then the port is added to the RADIUS-specified VLAN as a tagged port.
- If you configure dynamic VLAN assignment on a multi-device port authentication enabled interface, and the Access-Accept message returned by the RADIUS server does not contain a Tunnel-Private-Group-ID attribute, then it is considered an authentication failure, and the configured authentication failure action is performed for the MAC address.
- If the <vlan-name> string does not match either the name or the ID of a VLAN configured on the device, then it is considered an authentication failure, and the configured authentication failure action is performed for the MAC address.
- For tagged or dual-mode ports, if the VLAN ID provided by the RADIUS server does not match the VLAN ID in the tagged packet that contains the authenticated MAC address as its source address, then it is considered an authentication failure, and the configured authentication failure action is performed for the MAC address.
- If an untagged port had previously been assigned to a VLAN though dynamic VLAN assignment, and then another MAC address is authenticated on the same port, but the RADIUS Access-Accept message for the second MAC address specifies a different VLAN, then it is considered an authentication failure for the second MAC address, and the configured authentication failure action is performed. Note that this applies only if the first MAC address has not yet aged out. If the first MAC address has aged out, then dynamic VLAN assignment would work as expected for the second MAC address.

Specifying to Which VLAN a Port Is Moved After Its RADIUS-Specified VLAN Assignment Expires

When a port is dynamically assigned to a VLAN through the authentication of a MAC address, and the MAC session for that address is deleted on the Foundry device, then by default the port is removed from its RADIUS-assigned VLAN and placed back in the VLAN where it was originally assigned.

A port can be removed from its RADIUS-assigned VLAN when any of the following occur:

- The link goes down for the port

- The MAC session is manually deleted with the **mac-authentication clear-mac-session** command
- The MAC address that caused the port to be dynamically assigned to a VLAN ages out

For example, say port 1/1 is currently in VLAN 100, to which it was assigned when MAC address 0007.eaa1.e90f was authenticated by a RADIUS server. The port was originally configured to be in VLAN 111. If the MAC session for address 0007.eaa1.e90f is deleted, then port 1/1 is moved from VLAN 100 back into VLAN 111.

You can optionally specify an alternate VLAN to which to move the port when the MAC session for the address is deleted. For example, to place the port in the restricted VLAN, enter commands such as the following:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-auth move-back-to-old-vlan port-restrict-vlan
```

Syntax: [no] mac-authentication move-back-to-old-vlan disable | port-configured-vlan | system-default-vlan

The **disable** keyword disables moving the port back to its original VLAN. The port would stay in its RADIUS-assigned VLAN.

The **port-configured-vlan** keyword removes the port from its RADIUS-assigned VLAN and places it back in the VLAN where it was originally assigned. This is the default.

The **port-restrict-vlan** keyword removes the port from its RADIUS-assigned VLAN and places it in the restricted VLAN.

The **system-default-vlan** keyword removes the port from its RADIUS-assigned VLAN and places it in the DEFAULT-VLAN.

Saving Dynamic VLAN Assignments to the Running-Config File

You can configure the Foundry device to save the RADIUS-specified VLAN assignments to the device's running-config file. To do this, enter the following command:

```
BigIron(config)# mac-authentication save-dynamicvlan-to-config
```

Syntax: [no] mac-authentication save-dynamicvlan-to-config

By default, the dynamic VLAN assignments are not saved to the running-config file. Entering the **show running-config** command does not display dynamic VLAN assignments, although they can be displayed with the **show vlan** and **show auth-mac-address detail** commands.

Enabling Denial of Service Attack Protection

NOTE: This option is not available on the BigIron MG8 and NetIron 40G.

The Foundry device does not start forwarding traffic from an authenticated MAC address in hardware until the RADIUS server authenticates the MAC address; traffic from the non-authenticated MAC addresses is sent to the CPU. A denial of service (DoS) attack could be launched against the device where a high volume of new source MAC addresses is sent to the device, causing the CPU to be overwhelmed with performing RADIUS authentication for these MAC addresses. In addition, the high CPU usage in such an attack could prevent the RADIUS response from reaching the CPU in time, causing the device to make additional authentication attempts.

To limit the susceptibility of the Foundry device to such attacks, you can configure the device to use multiple RADIUS servers, which can share the load when there are a large number of MAC addresses that need to be authenticated. The Foundry device can run a maximum of 10 RADIUS clients per server and will attempt to authenticate with a new RADIUS server if current one times out.

In addition, you can configure the Foundry device to limit the rate of authentication attempts sent to the RADIUS server. When the multi-device port authentication feature is enabled, it keeps track of the number of RADIUS authentication attempts made per second. When you also enable the DoS protection feature, if the number of RADIUS authentication attempts for MAC addresses learned on an interface per second exceeds a configurable rate (by default 512 authentication attempts per second), the device considers this a possible DoS attack and disables the port. You must then manually re-enable the port.

The DoS protection feature is disabled by default. To enable it on an interface, enter commands such as the following:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication dos-protection enable
```

Syntax: [no] mac-authentication dos-protection enable

To specify a maximum rate for RADIUS authentication attempts, enter commands such as the following:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication dos-protection mac-limit 256
```

Syntax: [no] mac-authentication dos-protection mac-limit <number>

You can specify a rate from 1 – 65535 authentication attempts per second. The default is a rate of 512 authentication attempts per second.

Clearing Authenticated MAC Addresses

The Foundry device maintains an internal table of the authenticated MAC addresses (viewable with the **show authenticated-mac-address** command). You can clear the contents of the authenticated MAC address table either entirely, or just for the entries learned on a specified interface. In addition, you can clear the MAC session for an address learned on a specific interface.

To clear the entire contents of the authenticated MAC address table, enter the following command:

```
BigIron(config)# clear auth-mac-table
```

Syntax: clear auth-mac-table

To clear the authenticated MAC address table of entries learned on a specified interface, enter a command such as the following:

```
BigIron(config)# clear auth-mac-table e 3/1
```

Syntax: clear auth-mac-table <portnum>

To clear the MAC session for an address learned on a specific interface, enter commands such as the following:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication clear-mac-session 00e0.1234.abd4
```

Syntax: mac-authentication clear-mac-session <mac-address>

This command removes the Layer 2 CAM entry created for the specified MAC address. If the Foundry device receives traffic from the MAC address again, the MAC address is authenticated again.

Disabling Aging for Authenticated MAC Addresses

MAC addresses that have been authenticated or denied by a RADIUS server are aged out if no traffic is received from the MAC address for a certain period of time.

- Authenticated MAC addresses or non-authenticated MAC addresses that have been placed in the restricted VLAN are aged out if no traffic is received from the MAC address over the device’s normal MAC aging interval.
- Non-authenticated MAC addresses that are blocked by the device are aged out if no traffic is received from the address over a fixed hardware aging period (70 seconds), plus a configurable software aging period. (See the next section for more information on configuring the software aging period).

You can optionally disable aging for MAC addresses subject to authentication, either for all MAC addresses or for those learned on a specified interface.

Globally Disabling Aging of MAC Addresses

On most devices, you can disable aging for all MAC addresses on all interfaces where multi-device port authentication has been enabled by entering the following command:

```
BigIron(config)# mac-authentication disable-aging
```

However, on BigIron MG8 and NetIron 40G, enter the following command:

```
BigIron MG8(config)#mac-authentication disable-aging denied-only
```

Syntax: mac-authentication disable-aging denied-only | permitted-only

Enter the command at the global or interface configuration level.

The **denied-only** parameter prevents denied sessions from being aged out, but ages out permitted sessions.

The **permitted-only** parameter prevents permitted (authenticated and restricted) sessions from being aged out and ages denied sessions.

Disabling the Aging of MAC Addresses on Interfaces

NOTE: The following commands are not available on the BigIron MG8 and NetIron 40G.

To disable aging for all MAC addresses subject to authentication on a specific interface where multi-device port authentication has been enabled, enter the command at the interface level. For example:

```
BigIron(config)# interface e 3/1
BigIron(config-if-e100-3/1)# mac-authentication disable-aging
```

Syntax: [no] mac-authentication disable-aging

Specifying the Aging Time for Blocked MAC Addresses

When the Foundry device is configured to drop traffic from non-authenticated MAC addresses, traffic from the blocked MAC addresses is dropped in hardware, without being sent to the CPU. A Layer 2 CAM entry is created that drops traffic from the blocked MAC address in hardware. If no traffic is received from the blocked MAC address for a certain amount of time, this Layer 2 CAM entry is aged out. If traffic is subsequently received from the MAC address, then an attempt can be made to authenticate the MAC address again.

Aging of the Layer 2 CAM entry for a blocked MAC address occurs in two phases, known as ***hardware aging*** and ***software aging***. The hardware aging period is fixed at 70 seconds and is non-configurable. The software aging time is configurable through the CLI.

Once the Foundry device stops receiving traffic from a blocked MAC address, the hardware aging begins and lasts for a fixed period of time. After the hardware aging period ends, the software aging period begins. The software aging period lasts for a configurable amount of time (by default 120 seconds). After the software aging period ends, the blocked MAC address ages out, and can be authenticated again if the Foundry device receives traffic from the MAC address.

To change the length of the software aging period for blocked MAC addresses, enter a command such as the following:

```
BigIron(config)# mac-authentication max-age 180
```

Syntax: [no] mac-authentication max-age <seconds>

You can specify from 1 – 65535 seconds. The default is 120 seconds.

Displaying Multi-Device Port Authentication Information

You can display the following information about the multi-device port authentication configuration:

- Information about authenticated MAC addresses
- Information about the multi-device port authentication configuration
- Authentication Information for a specific MAC address or port
- Multi-device port authentication settings and authenticated MAC addresses for each port where the multi-device port authentication feature is enabled

- The MAC addresses that have been successfully authenticated
- The MAC addresses for which authentication was not successful

Displaying Authenticated MAC Address Information

To display information about authenticated MAC addresses on the ports where the multi-device port authentication feature is enabled, enter the following command:

```
BigIron# show authenticated-mac-address
----------------------------------------------------------------------
Port           Vlan  Accepted MACs    Rejected MACs    Attempted-MACs
----------------------------------------------------------------------
1/18           100   1                  100                0
1/20           40    0                  0                  0
1/22           100   0                  0                  0
4/5             30     0                  0                    0
```

Syntax: show authenticated-mac-address

The following table describes the information displayed by the **show authenticated-mac-address** command.

Table 7.2: Output from the show authenticated-mac-address command

This Field...	Displays...
Port	The port number where the multi-device port authentication feature is enabled.
Vlan	The VLAN to which the port has been assigned.
Accepted MACs	The number of MAC addresses that have been successfully authenticated
Rejected MACs	The number of MAC addresses for which authentication has failed.
Attempted-MACs	The rate at which authentication attempts are made for MAC addresses.

Displaying Multi-Device Port Authentication Configuration Information

To display information about the multi-device port authentication configuration, enter the following command:

```
BigIron# show authenticated-mac-address configuration

Feature enabled                 : Yes
Number of Ports enabled         : 4
---------------------------------------------------------------------------
Port  Fail-Action     Fail-vlan   Dyn-vlan  MAC-filter
---------------------------------------------------------------------------
1/18  Block Traffic   1            No        No
1/20  Block Traffic   1            No        No
1/22  Block Traffic   1            No        Yes
4/5   Block Traffic     1             No         No
```

Syntax: show authenticated-mac-address configuration

The following table describes the information displayed by the **show authenticated-mac-address configuration** command.

Table 7.3: Output from the show authenticated-mac-address configuration command

This Field...	Displays...
Feature enabled	Whether the multi-device port authentication feature is enabled on the Foundry device.
Number of Ports enabled	The number of ports on which the multi-device port authentication feature is enabled.
Port	Information for each multi-device port authentication-enabled port.
Fail-Action	What happens to traffic from a MAC address for which RADIUS authentication has failed: either block the traffic or assign the MAC address to a restricted VLAN.
Fail-vlan	The restricted VLAN to which non-authenticated MAC addresses are assigned, if the Fail-Action is to assign the MAC address to a restricted VLAN.
Dyn-vlan	Whether RADIUS dynamic VLAN assignment is enabled for the port.
MAC-filter	Whether a MAC filter has been applied to this port to specify pre-authenticated MAC addresses.

Displaying Multi-Device Port Authentication Information for a Specific MAC Address or Port

To display authentication information for a specific MAC address or port, enter a command such as the following:

```
BigIron# show authenticated-mac-address 0007.e90f.eaa1
-------------------------------------------------------------------------------
MAC/IP Address                   Port        Vlan Authenticated Time   Age CAM
                                                                           Index
-------------------------------------------------------------------------------
0007.e90f.eaa1 : 25.25.25.25     1/18        100  Yes    00d01h10m06s 0   N/A
```

Syntax: show authenticated-mac-address <mac-address> | <ip-address> | <portnum>

The <ip-address> parameter lists the MAC address associated with the specified IP address.

The <portnum> parameter lists the MAC addresses on the specified port.

The following table describes the information displayed by the **show authenticated-mac-address** command for a specified MAC address or port.

Table 7.4: Output from the show authenticated-mac-address <address> command

This Field...	Displays...
MAC/IP Address	The MAC address for which information is displayed. If the packet for which multi-device port authentication was performed also contained an IP address, then the IP address is displayed as well.
Port	The port on which the MAC address was learned.

Table 7.4: Output from the show authenticated-mac-address <address> command (Continued)

This Field...	Displays...
Vlan	The VLAN to which the MAC address was assigned.
Authenticated	Whether the MAC address was authenticated.
Time	The time at which the MAC address was authenticated. If the clock is set on the Foundry device, then the actual date and time are displayed. If the clock has not been set, then the time is displayed relative to when the device was last restarted.
Age	The age of the MAC address entry in the authenticated MAC address list.
CAM Index	If the MAC address is blocked, the index entry for the Layer 2 CAM entry created for this MAC address. If the MAC address is not blocked, either through successful authentication or through being placed in the restricted VLAN, then “N/A” is displayed. If the hardware aging period has expired, then “ffff” is displayed for the MAC address during the software aging period.

Displaying Multi-Device Port Authentication Settings and Authenticated MAC Addresses

To display the multi-device port authentication settings and authenticated MAC addresses for a port where the feature is enabled, enter the following command:

```
BigIron# show authenticated-mac-address detail e 1/18

Port                                : 1/18
Dynamic-Vlan Assignment             : Disabled
RADIUS failure action               : Block Traffic
Override-restrict-vlan              : No
Vlan                                : 100 ( RADIUS assigned: No)
DOS attack protection               : Disabled
Accepted Mac Addresses              : 1
Rejected Mac Addresses              : 100
Authentication in progress          : 0
Authentication attempts             : 0
RADIUS timeouts                     : 61250
Aging of MAC-sessions               : Enabled
Max-Age of MAC-sessions             : 120 seconds
MAC Filter  applied                 : No
-------------------------------------------------------------------------------
MAC/IP Address                  RADIUS Server    Authenticated  Time   Age  CAM
                                                                            Index
-------------------------------------------------------------------------------
00e0.1234.abd4 : 0.0.0.0         25.25.25.20      No      00d06h32m50s  224  083d
00e0.1234.abd5 : 0.0.0.0         25.25.25.20      No      00d06h32m50s  225  0843
00e0.1234.abd6 : 0.0.0.0         25.25.25.20      No      00d06h34m10s  216  084f
00e0.1234.abd7 : 0.0.0.0         25.25.25.20      No      00d06h34m50s  212  0862
00e0.1234.abd0 : 0.0.0.0         25.25.25.20      No      00d06h34m10s  217  081d
00e0.1234.abd1 : 0.0.0.0         25.25.25.20      No      00d06h32m50s  223  083a
00e0.1234.abd2 : 0.0.0.0         25.25.25.20      No      00d06h38m50s  189  0813
00e0.1234.abd3 : 0.0.0.0         25.25.25.20      No      00d06h32m50s  225  0840
00e0.1234.abdc : 0.0.0.0          25.25.25.20       No     00d06h34m50s  211  0816
```

Syntax: show authenticated-mac-address [<portnum>]

Omitting the <portnum> parameter displays information for all interfaces where the multi-device port authentication feature is enabled.

The following table describes the information displayed by the **show authenticated-mac-address** command.

Table 7.5: Output from the show authenticated-mac-address command

This Field...	Displays...
Port	The port to which this information applies.
Dynamic-Vlan Assignment	Whether RADIUS dynamic VLAN assignment has been enabled for the port.
RADIUS failure action	What happens to traffic from a MAC address for which RADIUS authentication has failed: either block the traffic or assign the MAC address to a restricted VLAN.

Table 7.5: Output from the show authenticated-mac-address command (Continued)

This Field...	Displays...
No-override-restrict-vlan	Whether a port can be dynamically assigned to a VLAN specified by a RADIUS server, if the port had been previously placed in the restricted VLAN because a previous attempt at authenticating a MAC address on that port failed.
Vlan	The VLAN to which the port is assigned, and whether the port had been dynamically assigned to the VLAN by a RADIUS server.
DOS attack protection	Whether denial of service attack protection has been enabled for multi-device port authentication, limiting the rate of authentication attempts sent to the RADIUS server.
Accepted Mac Addresses	The number of MAC addresses that have been successfully authenticated.
Rejected Mac Addresses	The number of MAC addresses for which authentication has failed.
Authentication in progress	The number of MAC addresses for which authentication is pending. This is the number of MAC addresses for which an Access-Request message has been sent to the RADIUS server, and for which the RADIUS server has not yet sent an Access-Accept message.
Authentication attempts	The total number of authentication attempts made for MAC addresses on an interface, including pending authentication attempts.
RADIUS timeouts	The number of times the session between the Foundry device and the RADIUS server timed out.
Aging of MAC-sessions	Whether software aging of MAC addresses is enabled.
Max-Age of MAC-sessions	The configured software aging period.
MAC Filter applied	Whether a MAC filter has been applied to this port to specify pre-authenticated MAC addresses.
MAC/IP Address	The MAC addresses learned on the port. If the packet for which multi-device port authentication was performed also contained an IP address, then the IP address is displayed as well.
RADIUS Server	The IP address of the RADIUS server used for authenticating the MAC addresses.
Authenticated	Whether the MAC address has been authenticated by the RADIUS server.
Time	The time at which the MAC address was authenticated. If the clock is set on the Foundry device, then the actual date and time are displayed. If the clock has not been set, then the time is displayed relative to when the device was last restarted.
Age	The age of the MAC address entry in the authenticated MAC address list.

Table 7.5: Output from the show authenticated-mac-address command (Continued)

This Field...	Displays...
CAM Index	If the MAC address is blocked, the index entry for the Layer 2 CAM entry created for this MAC address. If the MAC address is not blocked, either through successful authentication or through being placed in the restricted VLAN, then “N/A” is displayed. If the hardware aging period has expired, then “ffff” is displayed for the MAC address during the software aging period.

Displaying the Authenticated MAC Addresses

To display the MAC addresses that have been successfully authenticated, enter the following command:

```
BigIron# show auth-mac-addresses authorized-mac
```

Syntax: show auth-mac-addresses authorized-mac

Displaying the Non-Authenticated MAC Addresses

To display the MAC addresses for which authentication was not successful, enter the following command:

```
BigIron# show auth-mac-addresses unauthorized-mac
```

Syntax: show auth-mac-addresses unauthorized-mac

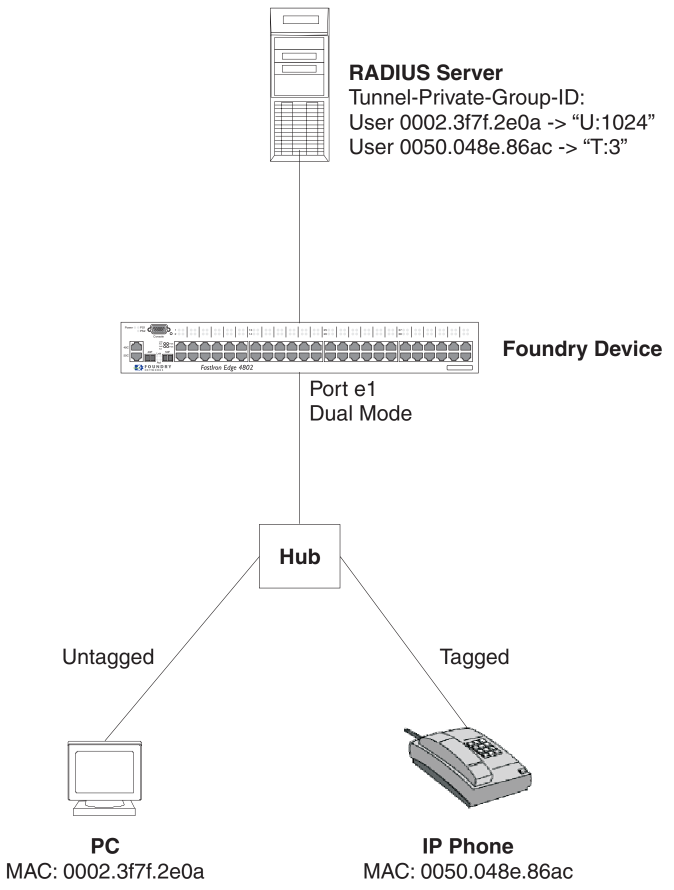
Sample Configurations

Multi-Device Port Authentication with Dynamic VLAN Assignment

NOTE: This configuration applies only to release 03.3.00 for the FES and release 02.2.00 for the FESX and FWSX.

Figure 7.1 illustrates multi-device port authentication with dynamic VLAN assignment on an FES. In this configuration, a PC and an IP phone are connected to a hub, which is connected to port e1 on an FES. Port e1 is configured as a dual-mode port. The PC transmits untagged traffic, and the IP phone is configured to transmit tagged traffic (VLAN 3). The profile for the PC’s MAC address on the RADIUS server specifies that the PC should be dynamically assigned to VLAN 1024, and the profile for the IP phone specifies that it should be dynamically assigned to VLAN 3.

Figure 7.1 Sample configuration using multi-device port authentication with dynamic VLAN assignment



In this example, multi-device port authentication is performed for both devices. If the PC is successfully authenticated, dual-mode port e1's PVID is changed from the VLAN 1 (the DEFAULT-VLAN) to VLAN 1024. If authentication for the PC fails, then the port's PVID can be changed to a specified "restricted" VLAN, or traffic from the PC can be blocked in hardware. In this sample configuration, the PVID for port e1 would be changed to that of the restricted VLAN, 1023.

If authentication for the IP phone is successful, then dual-mode port e1 is added to VLAN 3. If authentication for the IP phone fails, then traffic from the IP phone would be blocked in hardware. (Devices sending tagged traffic cannot be placed in the restricted VLAN.)

Prior to authentication, the part of the running-config related to multi-device port authentication would be as follows:

```
mac-authentication enable
mac-authentication auth-fail-vlan-id 1023

interface ethernet 1
 mac-authentication enable
 mac-authentication auth-fail-action restrict-vlan
 mac-authentication enable-dynamic-vlan
 dual-mode
```

After both devices were successfully authenticated, the **dual-mode** statement for port e1 would be changed to reflect the dynamically assigned PVID:

```
mac-authentication enable
mac-authentication auth-fail-vlan-id 1023
```

```
interface ethernet 1
 mac-authentication enable
 mac-authentication auth-fail-action restrict-vlan
 mac-authentication enable-dynamic-vlan
 dual-mode 1024
```

Had authentication for the PC failed, the VLAN ID in the **dual-mode** statement would be changed to 1023, the ID of the restricted VLAN.

Examples of Multi-Device Port Authentication and 802.1X Authentication Configuration on the Same Port (FES)

NOTE: This configuration examples below apply only to release 03.3.00 for the FES.

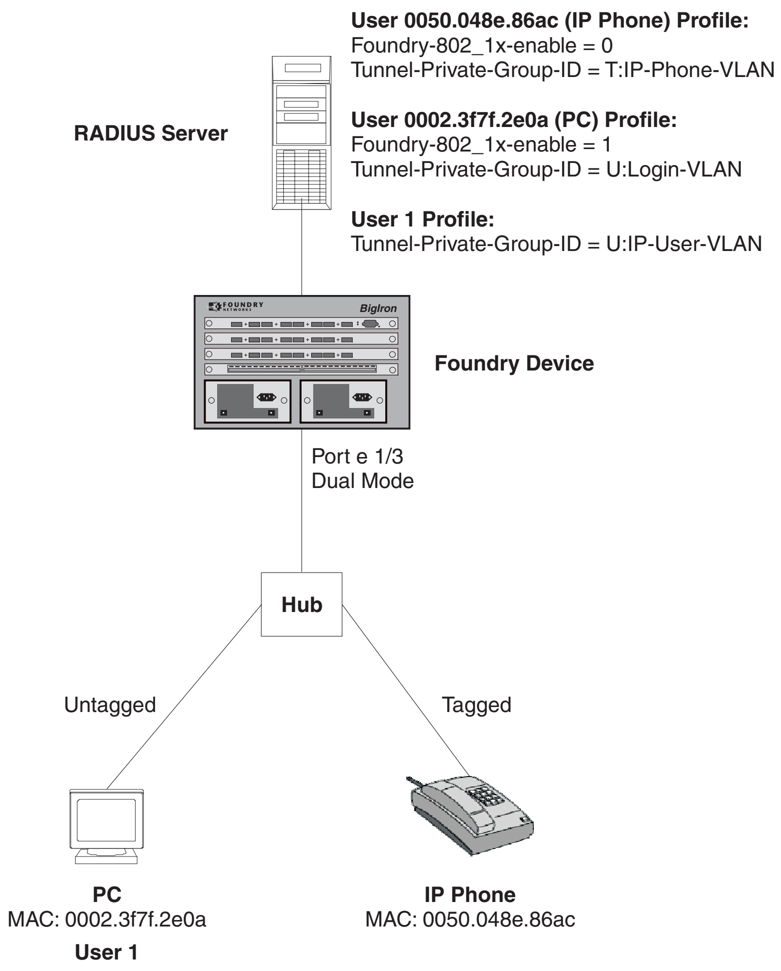
The following are two examples that use multi-device port authentication and 802.1X authentication on the same port.

Example 1

Figure 7.2 illustrates a sample configuration that uses multi-device port authentication and 802.1X authentication n the same port. In this configuration, a PC and an IP phone are connected to port e3 on an FES. Port e3 is configured as a dual-mode port.

The PC transmits untagged traffic, and the IP phone is configured to transmit tagged traffic (VLAN named "IP-Phone-VLAN"). The profile for the PC's MAC address on the RADIUS server specifies that the PC should be dynamically assigned to VLAN "Login-VLAN", and the profile for the IP phone specifies that it should be dynamically assigned to the VLAN named "IP-Phone-VLAN". When User 1 is successfully authenticated using 802.1X authentication, the PC is then placed in the VLAN named "User-VLAN".

Figure 7.2 Sample configuration using multi-device port authentication and 802.1X authentication on the same port


User 0050.048e.86ac (IP Phone) Profile:
Foundry-802_1x-enable = 0
Tunnel-Private-Group-ID = T:IP-Phone-VLAN
User 0002.3f7f.2e0a (PC) Profile:
Foundry-802_1x-enable = 1
Tunnel-Private-Group-ID = U:Login-VLAN
User 1 Profile:
Tunnel-Private-Group-ID = U:IP-User-VLAN
RADIUS Server
BigIron
Foundry Device
Port e 1/3
Dual Mode
Hub
Untagged
Tagged
PC
MAC: 0002.3f7f.2e0a
User 1
IP Phone
MAC: 0050.048e.86ac

When the devices attempt to connect to the network, they are first subject to multi-device port authentication.

When the IP phone's MAC address is authenticated, the Access-Accept message from the RADIUS server specifies that the IP phone's port be placed into the VLAN named "IP-Phone-VLAN". which is VLAN 7. The Foundry-802_1x-enable attribute is set to 0, meaning that 802.1X authentication is skipped for this MAC address. Port e3 is placed in VLAN 7 as a tagged port. No further authentication is performed.

When the PC's MAC address is authenticated, the Access-Accept message from the RADIUS server specifies that the PVID for the PC's port be changed to the VLAN named "Login-VLAN", which is VLAN 1024. The Foundry-802_1x-enable attribute is set to 1, meaning that 802.1X authentication is required for this MAC address. The PVID of the port e3 is temporarily changed to VLAN 1024, pending 802.1X authentication.

When User 1 attempts to connect to the network from the PC, he is subject to 802.1X authentication. If User 1 is successfully authenticated, the Access-Accept message from the RADIUS server specifies that the PVID for User 1's port be changed to the VLAN named "User-VLAN", which is VLAN 3. If 802.1X authentication for User 1 is unsuccessful, the PVID for port e3 is changed to that of the restricted VLAN, which is 1023, or untagged traffic from port e1 can be blocked in hardware.

Prior to authentication of the PC, the part of the running-config related to port e3 would be as follows:

```
interface ethernet 3
 dot1x port-control auto
 dual-mode
```

When the PC is authenticated using multi-device port authentication, the **dual-mode** statement for port e3 would be changed to reflect the dynamically assigned PVID of the “Login-VLAN”, which is VLAN 1024:

```
interface ethernet 3
 dot1x port-control auto
 dual-mode 1024
```

After User 1 is authenticated using 802.1X authentication, the **dual-mode** statement for port e3 would be changed to reflect the dynamically assigned PVID of the “User-VLAN”, which is VLAN 3:

```
interface ethernet 3
 dot1x port-control auto
 dual-mode 3
```

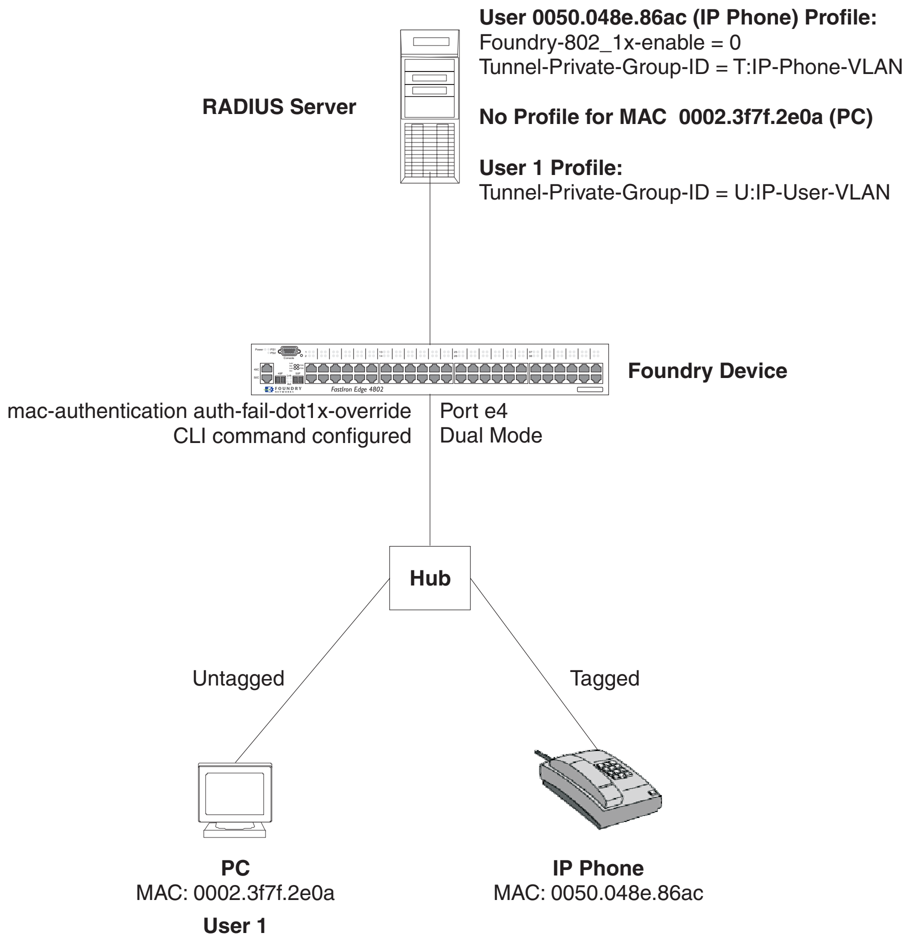
Example 2

The configuration in Figure 7.3 requires that you create a profile on the RADIUS server for each MAC address to which a device or user can connect to the network. In a large network, this can be difficult to implement and maintain.

As an alternative, you can create MAC address profiles only for those devices that do not support 802.1X authentication, such as IP phones and printers, and configure the FES to perform 802.1X authentication for the other devices that do not have MAC address profiles, such as user PCs. To do this, you configure the FES to perform 802.1X authentication when a device fails multi-device port authentication.

Figure 7.3 shows a configuration where multi-device port authentication is performed for an IP phone, and 802.1X authentication is performed for a user’s PC. There is a profile on the RADIUS server for the IP phone’s MAC address, but not for the PC’s MAC address.

Figure 7.3 Sample configuration where 802.1X authentication is performed when a device fails multi-device port authentication


User 0050.048e.86ac (IP Phone) Profile:
Foundry-802_1x-enable = 0
Tunnel-Private-Group-ID = T:IP-Phone-VLAN
RADIUS Server
No Profile for MAC 0002.3f7f.2e0a (PC)
User 1 Profile:
Tunnel-Private-Group-ID = U:IP-User-VLAN
Foundry Device
FastIron Edge 4802
mac-authentication auth-fail-dot1x-override
CLI command configured
Port e4
Dual Mode
Hub
Untagged
Tagged
PC
MAC: 0002.3f7f.2e0a
User 1
IP Phone
MAC: 0050.048e.86ac

Multi-device port authentication is initially performed for both devices. The IP phone's MAC address has a profile on the RADIUS server. This profile indicates that 802.1X authentication should be skipped for this device, and that the device's port be placed into the VLAN named "IP-Phone-VLAN".

Since there is no profile for the PC's MAC address on the RADIUS server, multi-device port authentication for this MAC address fails. Ordinarily, this would mean that the PVID for the port would be changed to that of the restricted VLAN, or untagged traffic on the port would be blocked in hardware. However, the FES is configured to perform 802.1X authentication when a device fails multi-device port authentication, so when User 1 attempts to connect to the network from the PC, he is subject to 802.1X authentication. If User 1 is successfully authenticated, the PVID for port e4 is changed to the VLAN named "User-VLAN".

To configure the FES to perform 802.1X authentication when a device fails multi-device port authentication, enter the following command:

```
FES4802 Router(config)# mac-authentication auth-fail-dot1x-override
```

Syntax: [no] mac-authentication auth-fail-dot1x-override

Chapter 8
Configuring Source IP Port Security

Overview

NOTE: This feature is supported on Foundry devices running Enterprise software release 07.8.00a or higher.

The ***source IP port security*** feature lets you configure interfaces on the Foundry device to allow traffic only from a specified number of “secure” IP addresses. Packets with source IP addresses that match the secure addresses are forwarded on the interface; packets from all other IP addresses are dropped.

The secure IP addresses can be learned two ways:

- Manually – You can specify secure IP addresses manually using the CLI. Manually specified addresses become secure immediately and remain secure as long as they remain in the device’s configuration.
- Dynamically – The Foundry device can learn secure IP addresses dynamically by examining address information in the control and data packets of various protocols, then validating the addresses.

  Once validated, dynamically learned addresses remain secure for a configurable time period. After this time period expires, the Foundry device sends an ARP request to the address. If a response is received from the connected host, the address remains secure for another time period. Otherwise, the address is aged out of the list of secure IP addresses.

By forwarding packets only from validated hosts, this feature can block traffic created by denial of service attacks that make use of “spoofed” or falsified source IP addresses.

The source IP port security feature is supported on JetCore Layer 2 Switchesonly. The feature is intended for use on ports on edge switches (or riser switches that feed off edge switches) that provide connectivity to end stations. It is not intended for uplink ports, or for ports that connect to routers.

The feature coexists with JetCore hardware-based ACLs configured on the device. As with JetCore ACLs, filtering of traffic for secure IP addresses is performed in hardware. In addition, the CLI for extended ACLs has been enhanced to allow you to configure ACLs that permit only packets whose source IP addresses are in the list of secure IP addresses for an interface.

Forwarding of traffic from validated source IP addresses takes precedence over any ACLs applied to the port. Once a host is in the list of validated IP addresses for a port, traffic from that host is forwarded regardless of whether ACLs are configured to deny traffic from the host.

The source IP port security feature is supported for trunk groups. When the feature is enabled on the primary port of a trunk, it is also enabled on the secondary ports. The secure IP addresses are learned on the primary port of the trunk. If the trunk is deleted dynamically, the feature is still enabled on all ports of the previously existing trunk, but the list of secure IP addresses is emptied, and then repopulated with new addresses by examining subsequent traffic.

NOTE: Source IP port security is supported on Ethernet interfaces only. It is not supported on POS or ATM interfaces.

How the Foundry Device Learns Secure IP Addresses

The secure IP addresses are learned in the following way:

- **Static configuration** – You can use the CLI to manually specify IP addresses to be made secure on a device or on an interface.
- **Dynamic learning** – The Foundry device learns and validates secure IP addresses by examining the following kinds of packets:
    - **ARP** – The device learns IP addresses from ARP requests. When an ARP response is received from a connected host, its IP address is added to the list of secure IP addresses for the interface.
    - **RARP and Dynamic RARP** – When a host is booted, it sends a RARP broadcast message containing its MAC address. The RARP server responds with an IP address for the host to use. The Foundry device examines the IP address in the RARP response and adds it to the list of secure IP addresses for the interface.
    - **DHCP** – The device examines DHCP packets sent to the CPU and interprets the ACK, RELEASE, and DECLINE messages to maintain addresses in the list of secure IP addresses for the interface
    - **BOOTP** – When a host is booted, it can send a BOOTP broadcast message containing the host's MAC address. The BOOTP server responds with the IP address for the MAC address. The device examines the BOOTP response and adds the IP address to the list of secure IP addresses for the interface.
    - **IP data packets** – The device examines the source IP address in IP data packets, enters the address into the secure IP address table with a status of LEARNING, then sends an ARP request to the source IP address. When an ARP response is received for the address, it is then considered validated, and the device changes the status of the IP address entry in the secure IP address table from LEARNING to ACTIVE.

The Foundry device places the secure IP addresses into an internal table of secure IP addresses. The statically configured secure IP addresses remain in the secure IP address table indefinitely, or until they are removed from the device's configuration. The dynamically learned secure IP addresses require protocol information to be maintained in the secure IP address table. They are refreshed by the protocol or are aged out after a configurable amount of time (by default, 180 seconds).

By default, each interface on which source IP port security is enabled can have a combined maximum of up to 64 statically configured or dynamically learned secure IP addresses. You can adjust the maximum number of secure IP addresses either globally or for a specific interface. When the source IP port security feature is enabled on an interface, all traffic is forwarded until the secure IP address table is fully populated with statically configured and dynamically learned secure IP addresses. Once the secure IP address table is fully populated, only packets with source IP addresses that match the addresses in the secure IP address table are forwarded on the interface. All other packets are dropped.

Attack Protection Feature

When source IP port security is enabled on an interface, the device validates IP addresses for incoming traffic and adds the IP addresses to the list of secure addresses until the configured maximum limit is reached. If an excessive amount of traffic is being received on the interface that has IP addresses that cannot be validated, it may be a sign of a denial of service attack against the Foundry device.

To detect such attacks and respond to them, you can enable the ***attack protection*** component of the source IP port security feature. When the attack protection feature is enabled, the Foundry device examines the number of attempts made to validate IP addresses over a specified interval. If that number exceeds a threshold value, then the Foundry device prevents any further IP addresses from being added to the list of secure addresses for that interface.

See "Configuring the Attack Protection Feature" on page 8-7 for more information.

Configuring Source IP Port Security

Configuring the source IP port security feature consists of the following tasks:

- Enabling the source IP port security feature
- Specifying the maximum number of secure IP addresses for an interface
- Specifying static secure IP addresses
- Specifying a trusted subnet for secure IP addresses
- Configuring dynamic learning of secure IP addresses
- Specifying the maximum age for dynamically learned secure IP addresses
- Disabling ARP probing when dynamically learned addresses reach the max-age value
- Enabling aging of dynamically learned secure IP addresses
- Configuring an ACL for secure IP addresses
- Configuring the attack protection feature

These tasks are described in the following sections. The tasks do not need to be completed in the order listed above. For example, you can specify static secure IP addresses prior to enabling the source IP port security feature on an Interface, although the CAM entries for the static secure IP addresses are not created until the feature is enabled.

Enabling Source IP Port Security

By default, the source IP port security feature is disabled. You can enable it either from the Global CONFIG level of the CLI or at the Interface level of the CLI.

To enable the feature on a single interface:

```
BigIron(config)# srcip-security enable e 1/1
```

To enable the feature on multiple interfaces:

```
BigIron(config)# srcip-security enable e 1/1 to 1/24
```

Syntax: srcip-security enable ethernet <slot/port> [to <slot/port>]

To enable the feature on a single interface:

```
BigIron(config)# int e 1/1
BigIron(config-if-e1000-1/1)# srcip-security enable
```

To enable the feature on multiple interfaces:

```
BigIron(config)# int e 1/1 to 1/24
BigIron(config-if-e1000-1/1-1/24)# srcip-security enable
```

Syntax: [no] srcip-security enable

NOTE: Use the **no** form of the command to disable the feature at the Interface level of the CLI. For instructions on disabling the feature at the Global CONFIG level of the CLI, see the following section.

Disabling Source IP Port Security at the Global CONFIG Level of the CLI

To disable the source IP port security feature from the Global CONFIG Level of the CLI, enter commands such as the following:

- To disable the feature on all currently active interfaces:

  ```
  BigIron(config)# srcip-security disable
  ```

- To disable the feature on an interface:

```
BigIron(config)# srcip-security disable e 1/1
```

- To disable the feature on multiple interfaces:

```
BigIron(config)# srcip-security disable e 1/1 to 1/24
```

Syntax: srcip-security disable [<slot/port> [to <slot/port>]]

Specifying the Maximum Number of Secure IP Addresses for an Interface

By default, each interface on which source IP port security is enabled can have up to 64 addresses in its list of secure IP addresses. You can adjust the maximum number of secure IP addresses either globally or for a specific interface.

The value specified for an interface takes precedence over the global maximum value. If the value specified for an interface is different from the global value, changing the global value does not change the value for the interface. For trunk groups, the maximum value specified for the primary port applies to the secondary ports.

To change the global maximum to 32 secure IP addresses, enter the following command:

```
BigIron(config)# srcip-security max-ipaddr-per-interface 32
```

Syntax: [no] srcip-security max-ipaddr-per-interface <number>

Enter a value from 1 – 128 for <number>. The default is 64.

After you enter this command, each interface on which the feature is enabled can have up to 32 addresses in its list of secure IP addresses.

To change the maximum number of secure IP addresses on interface 1/1 to 32, enter the following command:

```
BigIron(config)# int e 1/1
BigIron(config-if-e1000-1/1)# srcip-security max-ipaddr-per-interface 32
```

Syntax: [no] srcip-security max-ipaddr-per-interface <number>

NOTE: If you set the maximum number of secure IP addresses to be less than the number of secure IP address that are currently active on the device, then all addresses are removed from the secure IP address table, and addresses will be re-learned up to the new maximum number of secure IP addresses.

Specifying Static Secure IP Addresses

You can manually specify IP addresses that are to be made secure for an interface. Packets with the specified IP address and MAC address are always forwarded by the interface; they are never aged out of the interface’s list of secure IP addresses.

To specify a static secure IP address for an interface, enter commands such as the following:

```
BigIron(config)# int e 1/1
BigIron(config-if-e1000-1/1)# srcip-security static 192.168.9.210 0050.DA18.747C
```

Syntax: [no] srcip-security static <ip-address> <mac-address>

In this example, interface 1/1 always forwards packets that have a source IP address of 192.168.9.210 and MAC address 0050.DA18.747C.

Specifying a Trusted Subnet for Secure IP Addresses

You can optionally specify a trusted subnet for secure IP addresses. When a trusted subnet is specified, the secure IP addresses are validated against the subnet. Any IP address on this subnet is considered a secure IP address.

To specify a trusted subnet, enter the following command:

```
BigIron(config)# srcip-security trusted-net 192.168.0.0/16
```

Syntax: [no] srcip-security trusted-net <network-address>/<subnet-mask>

Up to six trusted subnets can be added.

Configuring Dynamic Learning of Secure IP Addresses

When you enable the source IP port security feature on an interface, dynamic learning of secure IP addresses is done by default. The device compiles a list of secure IP addresses by examining packets as described in “How the Foundry Device Learns Secure IP Addresses” on page 8-2. You can optionally disable dynamic learning of secure IP addresses globally or on a specific interface.

To disable dynamic learning of secure IP addresses for all interfaces on the Foundry device, enter the following command:

```
BigIron(config)# srcip-security no-dynamic-learning
```

To disable dynamic learning of secure IP addresses for interface 1/1, enter the following commands:

```
BigIron(config)# int e 1/1
BigIron(config-if-e1000-1/1)# srcip-security no-dynamic-learning
```

Syntax: [no] srcip-security no-dynamic-learning

When dynamic learning of secure IP addresses is disabled, the dynamically learned addresses are deleted from the system.

When no-dynamic-learning is disabled globally with the (**no srcip-security no-dynamic-learning** command), then the **srcip-security no-dynamic-learning** commands are removed from the individual ports where they had been configured.

NOTE: Use this command with caution. Under the following circumstances, the Foundry device could allow all traffic to be forwarded regardless of the source IP port security settings configured on the device:

- The source IP port security feature is enabled on an interface
- The total number of IP addresses in the secure IP address table that have either ACTIVE or VALIDATED status has not reached the limit specified with the **srcip-security max-ipaddr-per-interface** command.
- Dynamic learning of secure IP addresses is then disabled with the **srcip-security no-dynamic-learning** command

If all of these circumstances are true, then the device will forward all traffic without regard to source IP port security settings. To avoid this situation, make sure that the secure IP address table is populated to the limit specified with the **srcip-security max-ipaddr-per-interface** command before disabling dynamic learning of secure IP addresses.

Specifying the Maximum Age for Dynamically Learned Secure IP Addresses

Static secure IP addresses are never aged out of the list of secure IP addresses for an interface. By default, dynamically learned secure IP addresses are not aged out either, but they can be if you configure the **srcip-security age enable** command. See “Enabling Aging of Dynamically Learned Secure IP Addresses” on page 8-6.

When aging is enabled for for dynamically learned secure IP addresses, they can be aged out after a specified amount of time (by default, 180 seconds). You can set the aging time for dynamically learned secure IP addresses. This setting applies to all ports where the feature is enabled.

For example, to set the aging time for dynamically learned secure addresses to 60 seconds, enter the following command:

```
BigIron(config)# srcip-security max-age 60
```

Syntax: [no] srcip-security max-age <seconds>

You can set the aging time for dynamically learned secure IP addresses to between 10 – 65535 seconds. The default is 180 seconds.

Disabling ARP Probing When Dynamically Learned Addresses Reach the max-age Value

When dynamically learned secure IP addresses are validated, they remain secure for the time period configured with the **srcip-security max-age** command. After this time period expires, the Foundry device sends an ARP request to the address. If a response is received from the connected host, the address remains secure for another time period, after which the Foundry device sends another ARP request to the address, and so on. If an ARP response is not received from the host, then the address is aged out of the list of secure IP addresses.

You can optionally prevent the device from sending ARP requests to dynamically learned secure IP addresses when they reach the aging time configured with the **srcip-security max-age** command. To do so, enter the following command:

```
BigIron(config)# srcip-security no-probe-on-age
```

Syntax: [no] srcip-security no-probe-on-age

Enabling Aging of Dynamically Learned Secure IP Addresses

By default, aging of dynamically learned secure IP addresses is disabled so that dynamically learned secure IP addresses never age out. When aging is enabled, dynamically learned secure IP addresses are aged out of the list of secure IP addresses for the interface according to the **srcip-security max-age** setting described above.

To enable aging for dynamically learned secure addresses for all interfaces on the Foundry device, enter the following command:

```
BigIron(config)# srcip-security age enable
```

To enable aging for dynamically learned secure addresses for Ethernet interface 1/1, enter the following commands:

```
BigIron(config)# int e 1/1
BigIron(config-if-e1000-1/1)# srcip-security age enable
```

Syntax: [no] srcip-security age enable

Configuring an ACL for Secure IP Addresses

A new keyword, **secure-host**, has been added to the syntax for extended ACLs. This keyword allows you to configure an inbound ACL that filters only packets whose source IP addresses are in the list of secure IP addresses for an interface; the ACL denies packets from all other hosts. If any other outbound ACLs are applied to the interface, they operate as configured, without regard to secure IP addresses.

The following is an example of an ACL that uses the **secure-host** keyword:

```
BigIron(config)# access-list 181 permit ip secure-host host 192.168.5.10
BigIron(config)# access-list 181 deny ip any any

BigIron(config)# int e 1/1
BigIron(config-if-e1000-1/1)# ip access-group 181 in
```

The **secure-host** keyword serves as a placeholder for the <source-ip> parameter in an extended ACL entry. The **secure-host** keyword is a wildcard meaning “all validated secure IP addresses on the interface to which the extended ACL is applied”. Using the **secure-host** keyword allows you create an ACL for all the valid hosts connected to an Ethernet interface without having to manually configure an ACL entry for each host.

When an ACL that uses the **secure-host** keyword is applied to an Ethernet interface, the Foundry device dynamically creates ACL entries in the device's Layer 4 CAM for the secure IP addresses on the interface. In the example above, if IP addresses 10.10.10.1 and 10.10.10.2 are in the list of secure IP addresses for interface e 1/1, the Foundry device dynamically programs the following ACL entries in the Layer 4 CAM:

```
access-list 181 permit ip 10.10.10.1 host 192.168.5.10
access-list 181 permit ip 10.10.10.2 host 192.168.5.10
```

NOTE: When an ACL that uses the **secure-host** keyword is applied to an interface where the source IP port security feature is enabled, the ACL can be applied in the inbound direction only. You cannot apply an ACL that uses the **secure-host** keyword in the outbound direction.

NOTE: You can configure the the **secure-host** keyword for permit ACL entries only. The **secure-host** keyword is not currently supported for deny ACL entries.

Configuring the Attack Protection Feature

The attack protection feature allows you to set a threshold for the number of validation attempts that can be made on an interface where the source IP port security feature is enabled. If the number of validation attempts exceeds the threshold, then the device takes action to prevent further addresses from being added to the list of secure addresses for the interface.

To configure the attack protection feature, you enable it and set the following parameters:

- **detection-trigger** – Determines the threshold for the attack protection feature. This parameter multiplied by the value of the **max-ipaddr-per-interface** parameter specifies the threshold for the number of times a Foundry device attempts to validate source IP addresses on an interface. If the number of validation attempts on an interface exceeds this threshold, the Foundry device considers it to be an attack.

  The default value for the **detection-trigger** parameter is 5. If the value specified by the **max-ipaddr-per-interface** parameter is 32, then it is considered to be an attack if the number of validation attempts over the polling interval exceeds 160.
- **polling-interval** – Specifies the time interval over which the device samples the number of validation attempts. The default interval is 10 seconds. If the attack protection threshold is exceeded over this interval, it is considered to be an attack.
- **retries** – Specifies the number of successive occurrences of positively detected attacks before an action is taken. The default is 3 retries. If an attack is detected over 3 consecutive polling intervals, then the device takes action to protect against the attack.
- **attack-recovery-time** – When the Foundry device takes action against an attack, it creates a Layer 4 CAM entry that drops packets from addresses other than those currently in the list of secure IP addresses. This Layer 4 CAM entry is removed after a configurable amount of time, set with the **attack-recovery-time** parameter. By default, the **attack recovery-time** parameter is set to 300 seconds. After the time specified with the **attack recovery-time** parameter expires, the device is considered “recovered” from the attack. The Layer 4 CAM entry is removed, and the device can again dynamically learn secure IP addresses.
- **max-successive-attacks** – After the device recovers from an attack, if a subsequent attack is detected within the amount of time specified with the **attack-recovery-time** parameter, then the attack is considered a “successive” attack. After three successive attacks, the current list of secure IP addresses is frozen, and the source IP port security feature must be re-enabled on the interface in order for new addresses to be added to the list of secure IP addresses. You specify the maximum number of attacks with the **max-successive-attacks** parameter.

Enabling the Attack Protection Feature on an Interface

By default, the attack protection feature is disabled. You can enable the attack protection feature on all valid interfaces on a module. First make sure the source IP port security feature has been enabled globally or on an interface, then enter the following command:

```
BigIron(config)# srcip-security attack protection enable e 3/11
```

Syntax: [no] srcip-security attack protection enable <portnum>

Specifying the Attack Protection Threshold

To specify the threshold for the attack protection feature, first make sure a value for the **max-ipaddr-per-interface** parameter has been defined. (See “Specifying the Maximum Number of Secure IP Addresses for an Interface” on page 8-4 for information on setting this parameter.) Then enter a command such as the following:

```
BigIron(config)# srcip-security attack protection detection-trigger 4
```

Syntax: [no] srcip-security attack protection detection-trigger <number>

The default value for the **detection-trigger** parameter is 5. You can enter 1 – 100. If the **max-ipaddr-per-interface** value is 32, then the threshold for the attack protection feature is 160 validation attempts over the polling interval.

Specifying the Polling Interval

To specify the polling interval for the attack protection feature, enter a command such as the following:

```
BigIron(config)# srcip-security attack protection polling-interval 5
```

Syntax: [no] srcip-security attack protection polling-interval <seconds>

The default value for the **polling-interval** parameter is 10 seconds. You can enter 1 – 300. If the attack protection threshold is exceeded during this interval, then the Foundry device considers it to be an attack.

Specifying the Number of Retries Before an Action is Taken

If an attack is detected over a consecutive number of polling intervals, then the device takes action against the attack. You can specify this number of polling intervals by entering a command such as the following:

```
BigIron(config)# srcip-security attack protection retries 4
```

Syntax: [no] srcip-security attack protection retries <number>

Enter 1 – 10. The default value for the retries parameter is 3. If an attack is detected over 3 consecutive polling intervals, then the Foundry device takes action to protect against the attack.

Specifying the Attack Recovery Time

When the Foundry device takes action against an attack, it creates a Layer 4 CAM entry that drops packets from addresses other than those currently in the list of secure IP addresses. This Layer 4 CAM entry is removed after a configurable amount of time, after which the Foundry device is considered “recovered” from the attack, and can again dynamically learn secure IP addresses.

To set the attack recovery time, enter a command such as the following:

```
BigIron(config)# srcip-security attack-protection auto-attack-recovery-time 120
```

Syntax: [no] srcip-security attack-protection auto-attack-recovery-time <seconds>

You can specify from 1 – 65535 seconds. The default is 300 seconds.

Specifying the Maximum Number of Attacks

After the Foundry device recovers from an attack, if a subsequent attack is detected within the amount of time specified with the **attack-recovery-time** parameter, then the attack is considered a “successive” attack. By default, after three successive attacks, the current list of secure IP addresses is frozen, and the source IP port security feature must be re-enabled on the interface in order for new addresses to be added to the list of secure IP addresses. You can specify the maximum number of attacks with the **max-successive-attacks** parameter.

For example, if the **max-successive-attacks** parameter is set to 2, and the attack recovery time is 30 seconds, then the following takes place when the device detects an attack:

1. The device creates a Layer 4 CAM entry denying traffic from all addresses except those in the list of secure IP addresses.
2. After 30 seconds, the Layer 4 CAM entry is removed.

3. If an attack is detected in the 30 seconds following the removal of the Layer 4 CAM entry, then it considered a successive attack. Step 1 and Step 2 are repeated until the number of successive attacks detected exceeds 2.
4. If the number of successive attacks detected exceeds 2, the Layer 4 CAM entry is not removed, and you must manually re-enable source IP port security on the interface.

The following command sets the maximum number of successive attacks at 5:

```
BigIron(config)# srcip-security attack-protection max-successive-attacks 5
```

Syntax: [no] srcip-security attack-protection max-successive-attacks <number>

You can specify from 1 – 65535 successive attacks. The default is 3 successive attacks.

Specifying the maximum number of spoofed IP addresses that can be logged

You can limit the number of spoofed IP address that are logged by entering a command such as the following:

```
BigIron(config)# srcip-security log-limit 1000
```

Syntax: srcip-security log-limit <number>

Enter 1 – 65535 for <number>. The default is 10.

Displaying Secure IP Address Information

You can display a list of the secure IP addresses that the device has learned, as well as information about the secure IP address configuration on the device.

Displaying Secure IP Addresses

To display a list of the secure IP addresses that have been learned on the device, enter the following command:

```
BigIron# show srcip-sec-table
Total entries = 6  VALIDATED entries = 6 Active entries = 5
IP Address       MAC Address    Port State       Learn-Scheme Age
-------------    --- -------    ---- -----       ------------ ---
25.25.25.21      0004.8017.6300 2/1  ACTIVE      IP_DATA      20
25.25.25.25      0007.e90f.eaa1 2/1  ACTIVE      IP_DATA      20
1.1.1.4          0050.da18.747C 2/1  ACTIVE      STATIC       0
1.1.1.3          0050.da18.747B 2/1  ACTIVE      STATIC       0
192.168.150.1    0004.8080.34a8 2/2  ACTIVE      STATIC       0
192.168.150.111  0004.8080.3408 2/2  VALIDATED      STATIC        0
```

Syntax: show srcip-sec-table [<ip-address> | <portnum>]

To display entries for the entire device, omit the IP address or port number. Enter an <ip-address> to display information for a specific IP address. Enter a port number to display entries for a specific port.

The following table lists the output of the **show srcip-sec-table** command.

Table 8.1: Output of the show srcip-sec-table command

This field	Displays
Total entries	The total number of addresses in the secure IP address table for all interfaces on the device.
VALIDATED entries	The number of addresses that have been validated but not yet programmed in CAM.

Table 8.1: Output of the show srcip-sec-table command

This field	Displays
Active entries	The number of addresses on the port that have been validated and programmed in CAM.
IP Address	The secure IP addresses learned or in the process of being learned for the port.
MAC Address	The MAC addresses for the secure IP addresses.
Port	The port to which the device is connected.
State	The learning state for the address entry. This can be one of the following: LEARNED – The source IP and MAC addresses have been learned from the packet, but have not yet been validated through ARP probing. VALIDATED – The address entry has been successfully validated but not yet programmed in CAM. This can happen if the address limit specified with the **srcip-security max-ipaddr-per-interface** command is less than the number of configured static secure IP addresses, in which case only the number of static entries up to the configured address limit are programmed in CAM. The remaining entries remain in the VALIDATED state, but are not programmed in CAM; if the address limit is increased, these entries are programmed in CAM. Since static secure IP addresses are not validated through ARP probing, they initially have VALIDATED state (not LEARNED state). ACTIVE – The address entry has been validated and programmed in CAM.
Learn-Scheme	How the secure IP address was learned. This can be DHCP, IP_DATA, STATIC, ARP, or RARP.
Age	The age of the secure IP address.

Displaying Source IP Port Security Configuration information

To display information about the configuration for source IP port security, enter the following command:

```
BigIron# show srcip brief
Global Configuration:-
        Total Ports Enabled                  :    2
        Max no entries per interface (global):    64
        Age out time(seconds)                :    180
        Aging                                :    Not Active
        Log limit                            :    10
        Attack Protection                    :    Enabled
        Attack Detection Polling Interval    :    10
        Attack Detection Retries             :    3
        Attack Detection Trigger-factor      :    5
        Auto Attack Recovery Time            :    30 (seconds)
        Max successive attacks before
        permanently blocking IP traffic      :    3

Total entries = 5  VALIDATED entries = 5 Active entries = 05
port        Max   Max  addr   Aging  Dyn   learn    Forced
            hosts age  active        Learn attempts discards
----        ----  ---  ------ ------ ----- -------- --------
1/2          0     180  0      Dis     Ena   0        1
1/3           0     180   0      Dis     Ena    0        1
```

You can display information about secure IP addresses on individual interfaces or a range of interfaces. For example, to display source IP port security information for interface 1/3, enter the following command.

```
BigIron# show srcip brief e 1/3
Total entries = 5  VALIDATED entries = 5 Active entries = 5
port        Max   Max  addr   Aging  Dyn   learn    Forced
            hosts age  active        Learn attempts discards
----        ----  ---  ------ ------ ----- -------- --------
1/3          0     180    0      Dis     Ena    0         1
```

Syntax: show srcip brief [<portnum> [to <portnum>]]

The following table lists the output of the **show srcip brief** command.

Table 8.2: Output of the show srcip brief command

This field	Displays
Total Ports Enabled	The number of interfaces on which the source IP port security feature is enabled.
Max no entries per interface (global)	The maximum number of secure IP addresses that can be learned for an interface
Age out time(seconds)	If aging is enabled for dynamically learned secure IP addresses, the length of time before an address is aged out of the interface's list of secure IP addresses.
Aging	Whether aging is enabled or disabled for dynamically learned secure IP addresses.

Table 8.2: Output of the show srcip brief command

This field	Displays
Log limit	The number of times packets from spoofed IP addresses are processed by the Foundry device's CPU before the device generates a Syslog message.
Attack Protection	Whether the attack protection feature is enabled or disabled.
Attack Detection Polling Interval	The configured polling interval for the attack protection feature.
Attack Detection Retries	The number consecutive polling intervals in which an attack is detected before the device takes action against the attack.
Attack Detection Trigger-factor	The value, when multiplied by the value of the **max-ipaddr-per-interface** parameter, that indicates the threshold for the attack protection feature.
Auto Attack Recovery Time	The number of seconds the device freezes the list of secure IP addresses when an attack is detected.
Max successive attacks before permanently blocking IP traffic	The number of attacks that can occur before the list of secure IP addresses is frozen permanently.
Total entries	The total number of addresses in the secure IP address table for all interfaces on the device.
VALIDATED entries	The number of addresses that have been validated but not yet programmed in CAM.
Active entries	The number of addresses on the port that have been validated and programmed in CAM.
port	The interface on which source IP port security is enabled.
Max-hosts	The maximum number of secure IP addresses that can be learned on this port.
Max-age	The aging time for dynamically learned secure IP addresses.
addr-active	The number of address entries for the port with ACTIVE status in the secure IP address table.
Aging	Whether aging is enabled for dynamically learned secure IP addresses.
Dyn-Learn	Whether dynamic learning of secure IP addresses is enabled on the port.
Forced Discards	The number of times the device detected and prevented a spoofed IP attack on the port.

Clearing Secure IP Addresses

To clear the list of secure IP addresses for all interfaces, enter the following command:

```
BigIron# clear srcip-security
```

To clear a specified address from the list of secure IP addresses, enter a command such as the following:

```
BigIron# clear srcip-security 192.168.20.177
```

To clear the list of secure IP addresses for interface 1/1, enter the following commands:

```
BigIron# clear srcip-security ethernet 1/1
```

Syntax: clear srcip-security [<ip-address> | <portnum>]

Chapter 9
Protecting Against Denial of Service Attacks

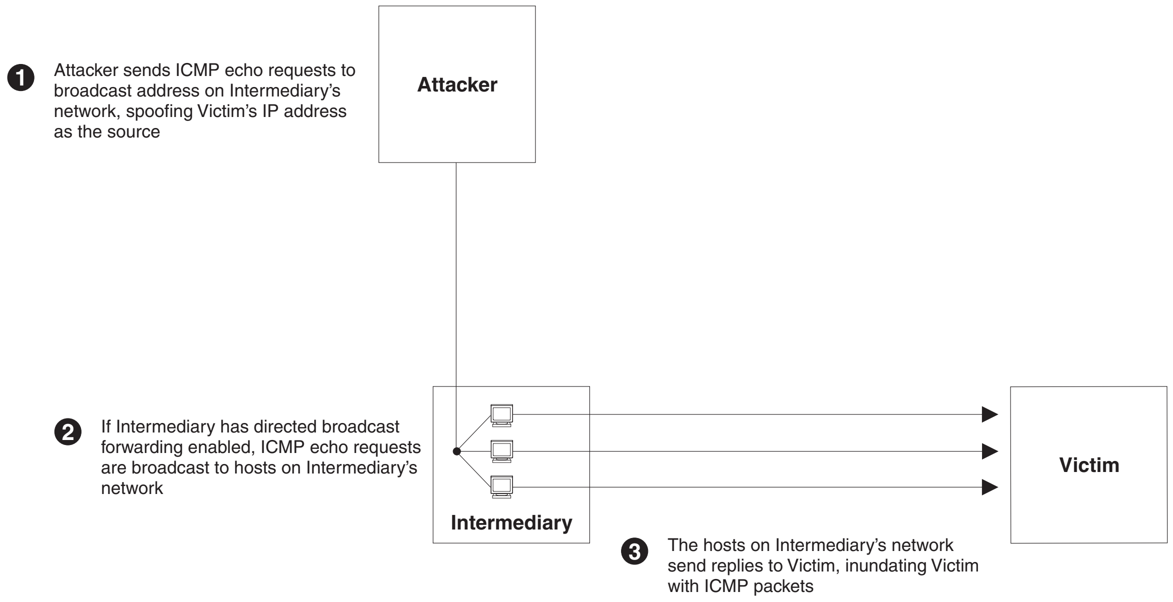
In a Denial of Service (DoS) attack, a router is flooded with useless packets, hindering normal operation. Foundry devices include measures for defending against two types of DoS attacks: Smurf attacks and TCP SYN attacks.

NOTE: For information about configuring the ServerIron to defend against TCP SYN attacks, see “ServerIron DoS Attack Protection” on page 10-1.

Protecting Against Smurf Attacks

A ***Smurf attack*** is a kind of DoS attack where an attacker causes a victim to be flooded with ICMP echo (Ping) replies sent from another network. Figure 9.1 illustrates how a Smurf attack works.

Figure 9.1 How a Smurf attack floods a victim with ICMP replies


1 Attacker sends ICMP echo requests to broadcast address on Intermediary’s network, spoofing Victim’s IP address as the source
Attacker
2 If Intermediary has directed broadcast forwarding enabled, ICMP echo requests are broadcast to hosts on Intermediary’s network
Intermediary
Victim
3 The hosts on Intermediary’s network send replies to Victim, inundating Victim with ICMP packets

The attacker sends an ICMP echo request packet to the broadcast address of an intermediary network. The ICMP echo request packet contains the spoofed address of a victim network as its source. When the ICMP echo request reaches the intermediary network, it is converted to a Layer 2 broadcast and sent to the hosts on the intermediary network. The hosts on the intermediary network then send ICMP replies to the victim network.

For each ICMP echo request packet sent by the attacker, a number of ICMP replies equal to the number of hosts on the intermediary network are sent to the victim. If the attacker generates a large volume of ICMP echo request packets, and the intermediary network contains a large number of hosts, the victim can be overwhelmed with ICMP replies.

Avoiding Being an Intermediary in a Smurf Attack

A Smurf attack relies on the intermediary to broadcast ICMP echo request packets to hosts on a target subnet. When the ICMP echo request packet arrives at the target subnet, it is converted to a Layer 2 broadcast and sent to the connected hosts. This conversion takes place only when directed broadcast forwarding is enabled on the device.

To avoid being an intermediary in a Smurf attack, make sure forwarding of directed broadcasts is disabled on the Foundry device. Starting with release 06.0.00, directed broadcast forwarding is disabled by default. In releases prior to 06.0.00, directed broadcast forwarding is enabled by default. To disable directed broadcast forwarding, do one of the following:

USING THE CLI

```
BigIron(config)# no ip directed-broadcast
```

Syntax: [no] ip directed-broadcast

USING THE WEB MANAGEMENT INTERFACE

1. Log on to the device using a valid user name and password for read-write access. The System configuration panel is displayed.
2. Click on the plus sign next to Configure in the tree view to display the list of configuration options.
3. Click on the plus sign next to IP to display the list of IP configuration options.
4. Select the General link to display the IP configuration panel.
5. Select Disable next to Directed Broadcast Forward.
6. Click the Apply button to save the change to the device's running-config file.
7. Select the Save link at the bottom of the dialog. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

Avoiding Being a Victim in a Smurf Attack

You can configure the Foundry device to drop ICMP packets when excessive numbers are encountered, as is the case when the device is the victim of a Smurf attack. You can set threshold values for ICMP packets that are targeted at the router itself or passing through an interface, and drop them when the thresholds are exceeded.

For example, to set threshold values for ICMP packets targeted at the router, enter the following command in CONFIG mode:

```
BigIron(config)# ip icmp burst-normal 5000 burst-max 10000 lockup 300
```

To set threshold values for ICMP packets received on interface 3/11:

```
BigIron(config)# int e 3/11
BigIron(config-if-e100-3/11)# ip icmp burst-normal 5000 burst-max 10000 lockup 300
```

Syntax: ip icmp burst-normal <value> burst-max <value> lockup <seconds>

The **burst-normal** value can be from 1 – 100000.

The **burst-max** value can be from 1 – 100000.

The **lockup** value can be from 1 – 10000.

This command is supported on Ethernet, POS, and Layer 3 ATM interfaces.

The number of incoming ICMP packets per second are measured and compared to the threshold values as follows:

- If the number of ICMP packets exceeds the **burst-normal** value, the excess ICMP packets are dropped.
- If the number of ICMP packets exceeds the **burst-max** value, *all* ICMP packets are dropped for the number of seconds specified by the **lockup** value. When the lockup period expires, the packet counter is reset and measurement is restarted.

In the example above, if the number of ICMP packets received per second exceeds 5,000, the excess packets are dropped. If the number of ICMP packets received per second exceeds 10,000, the device drops all ICMP packets for the next 300 seconds (five minutes).

Protecting Against TCP SYN Attacks

TCP SYN attacks exploit the process of how TCP connections are established in order to disrupt normal traffic flow. When a TCP connection starts, the connecting host first sends a TCP SYN packet to the destination host. The destination host responds with a SYN ACK packet, and the connecting host sends back an ACK packet. This process, known as a “TCP three-way handshake”, establishes the TCP connection.

While waiting for the connecting host to send an ACK packet, the destination host keeps track of the as-yet incomplete TCP connection in a connection queue. When the ACK packet is received, information about the connection is removed from the connection queue. Usually there is not much time between the destination host sending a SYN ACK packet and the source host sending an ACK packet, so the connection queue clears quickly.

In a TCP SYN attack, an attacker floods a host with TCP SYN packets that have random source IP addresses. For each of these TCP SYN packets, the destination host responds with a SYN ACK packet and adds information to the connection queue. However, since the source host does not exist, no ACK packet is sent back to the destination host, and an entry remains in the connection queue until it ages out (after around a minute). If the attacker sends enough TCP SYN packets, the connection queue can fill up, and service can be denied to legitimate TCP connections.

To protect against TCP SYN attacks, you can configure the Foundry device to drop TCP SYN packets when excessive numbers are encountered. You can set threshold values for TCP SYN packets that are targeted at the router itself or passing through an interface, and drop them when the thresholds are exceeded.

For example, to set threshold values for TCP SYN packets targeted at the router, enter the following command in CONFIG mode:

```
BigIron(config)# ip tcp burst-normal 10 burst-max 100 lockup 300
```

To set threshold values for TCP SYN packets received on interface 3/11:

```
BigIron(config)# int e 3/11
BigIron(config-if-e100-3/11)# ip tcp burst-normal 10 burst-max 100 lockup 300
```

Syntax: ip tcp burst-normal <value> burst-max <value> lockup <seconds>

The **burst-normal** value can be from 1 – 100000.

The **burst-max** value can be from 1 – 100000.

The **lockup** value can be from 1 – 10000.

NOTE: The **ip tcp burst-normal** command is available at the global CONFIG level on both Chassis devices and Stackable devices. On Chassis devices, this command is available at the Interface level as well. On the FES devices, this command is available at both the global CONFIG level and the Interface level. This command is supported on Ethernet, POS, and Layer 3 ATM interfaces.

The number of incoming TCP SYN packets per second are measured and compared to the threshold values as follows:

- If the number of TCP SYN packets exceeds the **burst-normal** value, the excess TCP SYN packets are dropped.

- If the number of TCP SYN packets exceeds the **burst-max** value, *all* TCP SYN packets are dropped for the number of seconds specified by the **lockup** value. When the lockup period expires, the packet counter is reset and measurement is restarted.

In the example above, if the number of TCP SYN packets received per second exceeds 10, the excess packets are dropped. If the number of TCP SYN packets received per second exceeds 100, the device drops all TCP SYN packets for the next 300 seconds (five minutes).

TCP Security Enhancement

NOTE: This feature applies to release 07.6.06 for the BigIron and release 02.0.02 and later for the NetIron IMR 640.

Software releases 07.6.06 and later and Multi-Service IronWare release 02.0.02 and later provide a TCP security enhancement that improves upon the handling of TCP inbound segments. This enhancement eliminates or minimizes the possibility of a TCP reset attack, in which a perpetrator attempts to prematurely terminate an active TCP session, and a data injection attack, wherein an attacker injects or manipulates data in a TCP connection.

In both cases, the attack is blind, meaning the perpetrator does not have visibility into the content of the data stream between two devices, but blindly injects traffic. Also, the attacker does not see the direct effect, the continuing communications between the devices and the impact of the injected packet, but may see the indirect impact of a terminated or corrupted session.

The TCP security enhancement prevents and protects against the following three types of attacks:

- Blind TCP reset attack using the reset (RST) bit.
- Blind TCP reset attack using the synchronization (SYN) bit
- Blind TCP packet injection attack

The TCP security enhancement is automatically enabled. If necessary, you can disable this feature. See “Disabling the TCP Security Enhancement” on page 9-5.

Protecting Against a Blind TCP Reset Attack Using the RST Bit

In a blind TCP reset attack using the RST bit, a perpetrator attempts to guess the RST segments in order to prematurely terminate an active TCP session.

Prior software releases apply the following rules to the RST bit when receiving TCP segments:

- If the RST bit is set and the sequence number is outside the expected window, the Foundry device silently drops the segment.
- If the RST bit is set and the sequence number is within the acceptable range, the Foundry device resets the connection

To prevent a user from using the RST bit to reset a TCP connection, in software releases 07.6.06 and later and Multi-Service IronWare release 02.0.02 and later, the RST bit is subject to the following rules when receiving TCP segments:

- If the RST bit is set and the sequence number is outside the expected window, the Foundry device silently drops the segment.
- If the RST bit is exactly the next expected sequence number, the Foundry device resets the connection.
- If the RST bit is set and the sequence number does not exactly match the next expected sequence value, but is within the acceptable window, the Foundry device sends an acknowledgement.

This TCP security enhancement is enabled by default in software releases 07.6.06 and later and Multi-Service IronWare release 02.0.02 and later. To disable it, see “Disabling the TCP Security Enhancement” on page 9-5.

Protecting Against a Blind TCP Reset Attack Using the SYN Bit

In a blind TCP reset attack, a perpetrator attempts to guess the SYN bits to prematurely terminate an active TCP session.

Prior software releases apply the following rules to the SYN bit when receiving TCP segments:

- If the SYN bit is set and the sequence number is outside the expected window, the Foundry device sends an ACK back to the sender.
- If the SYN bit is set and the sequence number is acceptable, the Foundry device sends a RST segment to the peer.

To prevent a user from using the SYN bit to tear down a TCP connection, in software releases 07.6.06 and later and Multi-Service IronWare release 02.0.02 and later, the SYN bit is subject to the following rules when receiving TCP segments:

- If the SYN bit is set and the sequence number is outside the expected window, the Foundry device sends an acknowledgement (ACK) back to the peer.
- If the SYN bit is set and the sequence number is an exact match to the next expected sequence, the Foundry device sends an ACK segment to the peer. Before sending the ACK segment, the software subtracts one from the value being acknowledged.
- If the SYN bit is set and the sequence number is acceptable, the Foundry device sends an acknowledgement (ACK) segment to the peer.

The TCP security enhancement is enabled by default. To disable it, see “Disabling the TCP Security Enhancement” on page 9-5.

Protecting Against a Blind Injection Attack

In a blind TCP injection attack, a perpetrator tries to inject or manipulate data in a TCP connection.

To reduce the chances of a blind injection attack, software releases 07.6.06 and later and Multi-Service IronWare release 02.0.02 and later perform an additional check on all incoming TCP segments.

This TCP security enhancement is enabled by default. To disable it, see “Disabling the TCP Security Enhancement” on page 9-5.

Disabling the TCP Security Enhancement

The TCP security enhancement is automatically enabled. If necessary, you can disable this feature. When you disable this feature, the Foundry device reverts to the original behavior (i.e., processes TCP segments as in prior releases).

To disable the TCP security enhancement, enter the following command at the Global CONFIG level of the CLI:

```
BigIron(config)# no ip tcp tcp-security
```

To re-enable the TCP security enhancement once it has been enabled, enter the following command:

```
BigIron(config)# ip tcp tcp-security
```

Syntax: [no] ip tcp tcp-security

Displaying Statistics about Packets Dropped Because of DoS Attacks

To display information about ICMP and TCP SYN packets dropped because burst thresholds were exceeded:

```
BigIron(config)# show statistics dos-attack
---------------------------- Local Attack Statistics --------------------------
ICMP Drop Count    ICMP Block Count    SYN Drop Count    SYN Block Count
---------------    ----------------    --------------    ---------------
              0                   0                 0                  0
--------------------------- Transit Attack Statistics -------------------------
Port   ICMP Drop Count    ICMP Block Count    SYN Drop Count    SYN Block Count
-----  ---------------    ----------------    --------------    ---------------

3/11                 0                   0                 0                  0
```

Syntax: show statistics dos-attack

To clear statistics about ICMP and TCP SYN packets dropped because burst thresholds were exceeded:

```
BigIron(config)# clear statistics dos-attack
```

Syntax: clear statistics dos-attack

Protecting Against DoS Attacks for All Protocols and Applications

Foundry devices provide protection from denial of service (DoS) attacks for management protocols such as HTTP, SSH, Telnet, and SNMP. To do so, ACLs applied to ingress ports filter on these management and application protocols. However, this feature is limited in that it provides protection from denial of service attacks for management protocols and applications only. It does not provide protection for all protocols.

On devices running Enterprise software release 08.0.00, the CPU ACL feature allows you to configure the Foundry device to provide protection from denial of service attacks for all protocols and applications received on the device. To do so, extended ACLs applied to ingress ports filter on the protocols specified in ACL clauses. For example, you can use extended ACLs to filter on protocols such as SNTP, TFTP, BGP, and ICMP.

The CPU ACL feature filters incoming packets in both hardware (JetCore only) and software. Packets filtered in hardware travel faster and use less resources in comparison to packets sent to the CPU for processing. If the device does not have available hardware resources (Content Addressable Memory (CAM)), the system will filter the packets in software (send the packets to the CPU for processing).

On a Foundry Layer 2 switch, hardware filtering protects the device from remote intrusions, thereby preventing DoS flooding and performance degradation on the device. However, on a Layer 3 switch, each interface can potentially be configured with multiple management IP addresses, in which case, CAM resources can quickly be consumed for all destination IP addresses. In this case, you can restrict management access to interfaces by disabling the management IP address through CAM (hardware denial).

The following sections provide instructions and examples for configuring CPU ACLs on a Foundry Layer 2 and Layer 3 Switch.

Configuring CPU ACL on a Layer 2 Switch

On a Foundry Layer 2 switch, hardware filtering protects it from remote intrusions, thereby preventing DoS flooding and performance degradation on the device.

The following is an example of configuring a JetCore Layer 2 Switch to perform filtering for Telnet access.

```
FastIron(config)# vlan 3 by port
FastIron(config-vlan-3)# untagged ethe 3/1 to 3/5
FastIron(config-vlan-3)# exit

FastIron(config)# ip address 10.10.11.1 255.255.255.0
FastIron(config)# access-list 100 deny tcp 10.10.11.0/24 any eq telnet
FastIron(config)# ip receive access-list 100 vlan 3
```

In this example, a Layer 2 VLAN is configured as a remote-access management VLAN. The IP address configured for the Foundry device is also used for the management IP address of the Layer 2 VLAN. Hardware filtering of Telnet traffic is performed for all the ports in the management VLAN. If CAM resources are exhausted, ACL clauses that are not programmed in CAM will be filtered in software.

Syntax: [no] ip receive access-list <ext ACL num> vlan <vlan num> | all

where <ext ACL num> is an extended ACL number from 100 – 199. Note that extended named ACLs are not supported.

<vlan num> is a valid VLAN number. Either enter the VLAN number or **all** to apply the command globally, on the entire device.

Use the **no** form of the command to disable the CPU ACL feature.

You can display information about CPU ACL filtering with the **show cam l4** command. For example:

```
FastIron(config)# show cam l4 3/2
Sl Index       Src IP_Addr  SPort      Dest IP_Addr  DPort Prot Age    Out Port
 3 24577      10.10.11.0/24   Any      10.10.11.1/24    23  TCP dis     Discard
 3 24579      10.10.11.0/24   Any      10.10.11.1/24    23  TCP dis     Discard
 3 24581      10.10.11.0/24   Any      10.10.11.1/24    23  TCP dis     Discard
 3 24583      10.10.11.0/24   Any      10.10.11.1/24    23  TCP dis     Discard
```

The source IP address of the filter entry is taken from the extended ACL, and the destination IP address of the filter entry is always taken from the management IP address of the Layer 2 Switch, regardless of the destination IP address specified in the ACL clause.

NOTE: Specifying a port number with the **show cam l4** command, for example, **show cam l4 3/2**, causes the device to display Layer 4 CAM information for all of the ports in the same port region (DMA) as port 3/2.

Configuring CPU ACL on a Layer 3 Switch

NOTE: On a Foundry Layer 3 switch, each interface can potentially be configured with multiple management IP addresses, thus, CAM resources can quickly be consumed for all destination IP addresses. In this case, you can restrict management access to interfaces by disabling the management IP address through CAM (hardware denial). See "Disabling an Interface's Access to Management Functions" on page 9-8.

The following is an example of configuring a JetCore Layer 3 Switch to perform filtering for Telnet access.

```
BigIron(config)# vlan 3 by port
BigIron(config-vlan-3)# untagged ethe 3/1 to 3/5
BigIron(config-vlan-3)# exit

BigIron(config)# interface ve 3
BigIron(config-ve-3)# ip address 10.10.11.1 255.255.255.0
BigIron(config-ve-3)# exit

BigIron(config)# access-list 100 permit tcp host 10.10.11.254 any eq telnet
BigIron(config)# access-list 100 permit tcp host 192.168.2.254 any eq telnet
BigIron(config)# access-list 100 permit tcp host 192.168.12.254 any eq telnet
BigIron(config)# access-list 100 permit tcp host 192.64.22.254 any eq telnet
BigIron(config)# access-list 10 deny any
```

In this example, a Layer 3 VLAN is configured as a remote-access management VLAN and a router interface. The IP address specified for the router interface becomes the management IP address of the VLAN. Hardware filtering of Telnet traffic is performed for all the ports in the management VLAN. If CAM resources are exhausted, ACL clauses that are not programmed in CAM (hardware) will be filtered in software (sent to the CPU for processing).

When you make changes to the ACL configuration and/or make changes to the management VLAN, you must enter the following command after making the configuration changes:

```
BigIron(config)# remote-management rebind
```

Syntax: remote-management rebind

The **show cam l4** command displays the following information about the hardware filtering in this configuration:

```
BigIron Router(config)#show cam l4 3/1
Sl Index       Src IP_Addr  SPort      Dest IP_Addr  DPort Prot Age    Out Port
 3 40960   192.64.22.254/32   Any      10.10.11.1/24    23  TCP dis   Use L2/L3
 3 40962  192.168.12.254/32   Any      10.10.11.1/24    23  TCP dis   Use L2/L3
 3 40964   192.168.2.254/32   Any      10.10.11.1/24    23  TCP dis   Use L2/L3
 3 40966    10.10.11.254/32   Any      10.10.11.1/24    23  TCP dis   Use L2/L3
 3 40968                Any   Any      10.10.11.1/24    23  TCP dis     Discard
```

The IP address in extended ACL 100 is the source IP address of the filter entry, and the IP address of the router interface is always the destination IP address of the filter entry, regardless of the destination IP address specified in the ACL clause.

NOTE: Specifying a port number with the **show cam l4** command, for example, **show cam l4 3/2**, causes the device to display Layer 4 CAM information for all of the ports in the same port region (DMA) as port 3/2.

Example CPU ACL Configuration on a Layer 3 Switch

The following shows an example CPU ACL configuration.

```
BigIron(config)# ip receive access-list 100 vlan 3
BigIron(config)# access-list 100 deny icmp any any packet-too-big
BigIron(config)# access-list 100 permit icmp any any echo
BigIron(config)# access-list 100 permit udp 192.168.2/253/24 any eq tftp
BigIron(config)# access-list 100 permit udp 192.168.2.253/24 any eq snmp
BigIron(config)# access-list 100 permit udp 192.168.2.253/24 any eq sntp
BigIron(config)# access-list 100 permit tcp 192.168.2.253/24 any eq ftp
BigIron(config)# access-list 100 permit tcp 192.168.2.253/24 any eq http
BigIron(config)# access-list 100 permit tcp 192.168.2.253/24 any eq ssh
BigIron(config)# access-list 100 permit tcp 192.168.2.253/24 any eq telnet
BigIron(config)# access-list 100 permit ospf any any precedence internet
BigIron(config)# access-list 100 deny ip any any'
```

Disabling an Interface’s Access to Management Functions

You can protect the CPU from remote access to management and application protocols. To enable this feature, disable access to the Management IP address through the device’s Content Addressable Memory (CAM). The following shows an example configuration.

NOTE: This feature does not affect Layer 3 routing functions.

```
BigIron(config)# global-protocol-vlan
BigIron(config)# vlan 1 name DEFAULT-VLAN by port
BigIron(config-vlan-1)# exit

BigIron(config)# router ospf
BigIron(config-ospf-router)# area 0
BigIron(config-ospf-router)# exit

BigIron(config)# int e 3/10
BigIron(config-if-e1000-3/10)# ip address 10.10.10.1 255.255.255.0
BigIron(config-if-e1000-3/10)# ip ospf area 0
BigIron(config-if-e1000-3/10)# exit

BigIron(config)# int e 3/11
BigIron(config-if-e1000-3/11)# ip address 11.11.11.1 255.255.255.0
BigIron(config-if-e1000-3/11)# ip ospf area 0
BigIron(config-if-e1000-3/11)# management-ip-disable
BigIron(config-if-e1000-3/11)# exit

BigIron(config)# int e 3/12
BigIron(config-if-e1000-3/12)# ip address 12.12.12.1 255.255.255.0
BigIron(config-if-e1000-3/12)# ip ospf area 0
BigIron(config-if-e1000-3/12)# management-ip-disable
BigIron(config-if-e1000-3/12)# exit

BigIron(config)# int e 3/13
BigIron(config-if-e1000-3/13)# ip address 13.13.13.1 255.255.255.0
BigIron(config-if-e1000-3/13)# ip ospf area 0
BigIron(config-if-e1000-3/13)# management-ip-disable
BigIron(config-if-e1000-3/13)# exit
```

Syntax: [no] ip address <ip-addr> <ip-mask>

where <ip-addr> and <ip-mask> are the destination IP address and subnet mask.

Syntax: [no] management-ip-disable

Use the **no** form of the command to re-enable access to the Management IP address.

Viewing Information about Disabled Management IPs

Use the **show cam l4** command to display information about CAM entries for disabled Management IP addresses.

```
BigIron Router(config)#show cam l4 3/11

Sl Index       Src IP_Addr  SPort      Dest IP_Addr  DPort Prot Age    Out Port
 3 40960                Any   Any      11.11.11.1/24    23  TCP dis     Discard
 3 40962                Any   Any      11.11.11.1/24    80  TCP dis     Discard
 3 40964                Any   Any      11.11.11.1/24  1812  TCP dis     Discard
 3 40966                Any   Any      11.11.11.1/24    49  TCP dis     Discard
 3 40968                Any   Any      11.11.11.1/24    22  TCP dis     Discard
 3 40970                Any   Any      12.12.12.1/24    23  TCP dis     Discard
 3 40972                Any   Any      12.12.12.1/24    80  TCP dis     Discard
 3 40974                Any   Any      12.12.12.1/24  1812  TCP dis     Discard
 3 40976                Any   Any      12.12.12.1/24    49  TCP dis     Discard
 3 40978                Any   Any      12.12.12.1/24    22  TCP dis     Discard
 3 43520                Any   Any      11.11.11.1/24   161  UDP dis     Discard
 3 43522                Any   Any      11.11.11.1/24    69  UDP dis     Discard
 3 43524                Any   Any      11.11.11.1/24    49  UDP dis     Discard
 3 43526                Any   Any      12.12.12.1/24   161  UDP dis     Discard
 3 43528                Any   Any      12.12.12.1/24    69  UDP dis     Discard
 3 43530                Any   Any      12.12.12.1/24    49  UDP dis     Discard
```

See the *Foundry Switch and Router Command Line Interface Reference* for definitions of the fields shown in the display.

Chapter 10
ServerIron DoS Attack Protection

The ServerIron can be configured to defend against Smurf and TCP SYN Denial of Service (DoS) attacks. To defend against Smurf attacks, you can set threshold values for ICMP packets targeted at the ServerIron's management IP address. See "Avoiding Being a Victim in a Smurf Attack" on page 9-2. To defend against TCP SYN attacks, the ServerIron includes the following features:

- "TCP SYN Attack Protection", which allows you to set a threshold for the amount of time it takes for a connecting host to send back an TCP ACK packet. See the next section, "TCP SYN Attack Protection"
- SYN-Defense™, which configures the ServerIron to complete the TCP three-way handshake on behalf of a connecting client. See "SYN-Defense™" on page 10-4.
- SYN-Guard™, which configures the ServerIron to forward packets to the destination server only after a three-way handshake has been completed with the connecting client. See "SYN-Guard™" on page 10-4.

NOTE: The SYN-Defense and SYN-Guard features are available on ServerIron Chassis devices only.

TCP SYN Attack Protection

TCP SYN attacks exploit the process of how TCP connections are established in order to disrupt normal traffic flow. When a TCP connection starts, the connecting host first sends a TCP SYN packet to the destination host. The destination host (actually the ServerIron, acting as an intermediary between the source and destination hosts) responds with a SYN ACK packet. The connecting host then sends back an ACK packet. This process, known as a "TCP three-way handshake", establishes the TCP connection.

When the ServerIron sends a SYN ACK packet to the connecting host, it adds an entry to its session table (as does the destination host). This entry remains in the session table until the connection ends or ages out. Up to 1,000,000 sessions are supported on a ServerIron with 32MB memory installed, and up to 160,000 sessions are supported on a ServerIron with 8MB of memory.

In a TCP SYN attack, an attacker floods a host with TCP SYN packets. For each of these TCP SYN packets, the ServerIron responds with a SYN ACK packet and adds an entry to its session table. However, no ACK packet is ever sent back, so the connection is incomplete. If the attacker sends enough TCP SYN packets, the session table can fill up with incomplete connections, and service can be denied to legitimate TCP connections.

To protect against TCP SYN attacks, you can set a threshold for the amount of time it takes for a connecting host to send back an ACK packet. If this threshold is exceeded, the ServerIron removes the entry for the connection from its session table, and a TCP RESET packet is sent to the destination real server, causing it to remove the entry from its session table as well.

USING THE CLI

For example, to configure the ServerIron to remove an entry from its session table if the connection remains incomplete for 6 or more seconds:

```
ServerIron(config)# server syn-def 6
```

Syntax: server syn-def <threshold>

After the ServerIron sends a SYN ACK packet to the connecting host, if no ACK packet is returned within 6 seconds, the entry for the connection is removed from the ServerIron's session table, and a TCP RESET packet is sent to the destination real server.

You can specify a threshold of between 0 – 16 seconds. A threshold of 0 disables this feature. Foundry recommends a threshold above 5 seconds. The default threshold is 8 seconds.

To display information about the number of times the incomplete connection threshold was reached:

```
ServerIron# show server traffic
Client->Server      =          0  Server->Client      =          0
Drops               =          0  Aged                =          0
Fw_drops            =          0  Rev_drops           =          0
FIN_or_RST          =          0  old-conn            =          0
Disable_drop        =          0  Exceed_drop         =          0
Stale_drop          =          0  Unsuccessful        =          0
TCP SYN-DEF RST     =          0  Server Resets       =          0
Out of Memory       =          0  Out of Memory       =          0
```

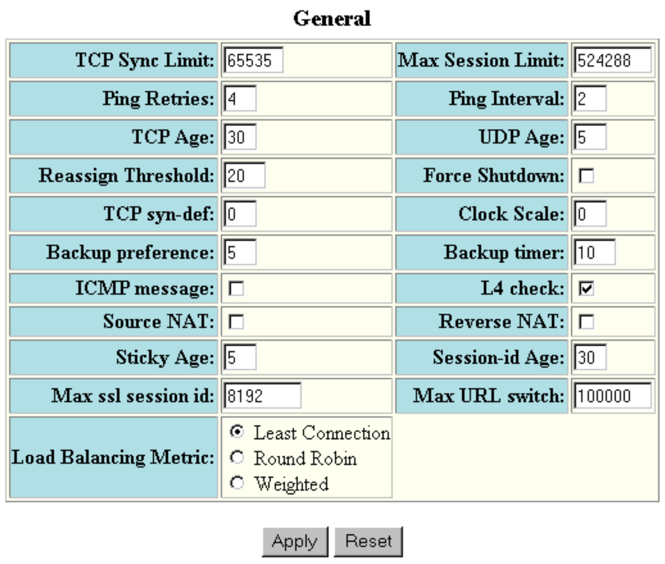
Syntax: show server traffic

The last line contains information relevant to the incomplete connection threshold. The TCP SYN-DEF RST field displays the number of times the incomplete connection threshold was reached. The Server Resets field displays the number of times the ServerIron sent a TCP RESET packet to the destination real server.

USING THE WEB MANAGEMENT INTERFACE

1. Log on to the device using a valid user name and password for read-write access.
2. Click on the plus sign next to Configure in the tree view to expand the list of system configuration options.
3. Click on the plus sign next to SLB in the tree view to expand the list of server load balancing option links.

4. Select the General link to display the following panel:



General

TCP Sync Limit:	65535	Max Session Limit:	524288
Ping Retries:	4	Ping Interval:	2
TCP Age:	30	UDP Age:	5
Reassign Threshold:	20	Force Shutdown:	☐
TCP syn-def:	0	Clock Scale:	0
Backup preference:	5	Backup timer:	10
ICMP message:	☐	L4 check:	☑
Source NAT:	☐	Reverse NAT:	☐
Sticky Age:	5	Session-id Age:	30
Max ssl session id:	8192	Max URL switch:	100000
Load Balancing Metric:	● Least Connection ○ Round Robin ○ Weighted		

Apply Reset

5. In the TCP syn def field, enter the threshold for incomplete connections in seconds. You can specify a threshold of between 0 – 16 seconds. A threshold of 0 disables this feature. Foundry recommends a threshold above 5 seconds. The default threshold is 8 seconds.
6. Click the Apply button to save the changes to the device's running-config file.
7. Select the Save link at the bottom of the panel. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

Configuring TCP SYN Protection on an Individual Port Basis

To configure TCP SYN protection for individual ports:

- Globally enable the feature and specify the number of seconds the ServerIron allows an incomplete TCP connection to remain in the session table.
- On individual ports, enable the feature. The number of seconds for incomplete connections that you specify globally applies to the individual ports on which you enable the TCP SYN protection feature. The feature applies only to the ports on which you enable it.

To configure TCP SYN protection, enter commands such as the following:

```
ServerIron(config)# server syn-def 6
ServerIron(config)# interface ethernet 1/1
ServerIron(config-if-1/1)# syn-def
ServerIron(config-if-1/1)# interface ethernet 1/3
ServerIron(config-if-1/3)# syn-def
ServerIron(config-if-1/3)# interface ethernet 4/8
ServerIron(config-if-4/8)# syn-def
```

The first command globally enables the feature and specifies 6 seconds as the maximum number of seconds an incomplete TCP connection can remain in the session table. The remaining commands enable the feature on ports 1/1, 1/3, and 4/8.

Syntax: server syn-def <seconds>

Syntax: syn-def

The <seconds> parameter specifies the TCP SYN threshold, which is the number of seconds an incomplete TCP connection can remain in the session table. You can specify from 0 – 16 seconds. If you specify 0, the feature is disabled. Foundry recommends a threshold above 5 seconds. There is no default.

SYN-Defense™

SYN-Defense provides an enhancement to the TCP SYN attack protection described in the previous section by allowing the ServerIron to complete the TCP three-way handshake on behalf of a connecting client.

When a connecting client sends a TCP SYN to a server, the ServerIron Chassis device forwards the SYN to the server, then sends a SYN ACK back to the client; this functionality is the same as it is on the other ServerIron models. However, unlike the other ServerIron models, the ServerIron Chassis device next sends an ACK to the real server, completing the three-way handshake on behalf of the connecting client. This allows the server to move the connection from its pending connection queue to its established connection queue, which is much larger.

If and when the client sends an ACK to the server, the ServerIron Chassis device passes it on to the server, which considers it to be a duplicate packet and drops it. If the client never sends an ACK to the server, the ServerIron Chassis device waits the number of seconds specified in the **server syn-def** command and then sends a TCP RESET packet to the server to reset the connection.

To configure TCP SYN attack protection on a ServerIron Chassis device, you specify the number of seconds the device waits for the client to send back an ACK packet, as well as the interface where TCP SYN attacks may be encountered. Unlike the other ServerIron models, which protect only SLB flows, the ServerIron Chassis devices protect all flows from TCP SYN attacks. A flow can be SLB, TCS, FWLB or just pass-through traffic.

For example, the following commands configure a ServerIron Chassis device to reset connections on port 3/1 when an ACK is not received from the client 6 seconds after a SYN ACK is sent.

```
ServerIron(config)# server syn-def 6
ServerIron(config)# interface e 3/1
ServerIron(config-if-3/1)# syn-def
```

Syntax: server syn-def <threshold>

Syntax: syn-def

NOTE: Do not configure the SYN-Defense feature simultaneously with the SYN-Guard feature (described below).

Disabling the SYN-Defense Three-Way Handshake

The SYN-Defense feature causes the ServerIron to send an ACK to the real server, completing the three-way handshake on behalf of the connecting client. This allows the real server to move the connection from its pending connection queue to its established connection queue, which is much larger.

Starting with Stackable release 07.3.07, you can prevent the ServerIron from sending the ACK to the real server to complete the three-way handshake. To do this, enter the following command:

```
ServerIron(config)# server syn-def-dont-send-ack
```

Syntax: server syn-def-dont-send-ack

SYN-Guard™

The SYN-Guard feature on ServerIron Chassis devices allows TCP connections to be terminated on the Foundry device. When this feature is enabled, the ServerIron completes the three-way handshake with a connecting client. Only when the three-way handshake is completed does the ServerIron establish a connection with the destination server and forward packets from the client to the server.

When a connecting client sends a TCP SYN to a server, the ServerIron responds with a SYN ACK and creates an internal session, but does not pass the SYN to the server. The ServerIron then waits for a specified amount of time for the client to send an ACK. If the client does not send an ACK within the allotted time, then the session is deleted. If the client does send an ACK, the ServerIron then establishes a session with the destination server. This allows the ServerIron to forward only packets associated with an established connection to the server. If the ServerIron cannot establish a session with the destination server within 8 seconds (the default), the ServerIron sends a TCP RESET to the client.

You enable the SYN-Guard feature on individual ports on ServerIron Chassis devices. This feature can be applied to inbound SYN requests (for Web site traffic) and/or outbound SYN requests (for ISP and institution outgoing traffic).

To activate the SYN-Guard feature and set the amount of time the ServerIron Chassis device waits for a client to send an ACK:

```
ServerIron(config)# ip tcp syn-proxy 12
```

Syntax: ip tcp syn-proxy <threshold>

To use the SYN-Guard feature for inbound SYN requests on interface 3/1:

```
ServerIron(config)# interface e 3/1
ServerIron(config-if-3/1)# ip tcp syn-proxy in
```

To use the SYN-Guard feature for outbound SYN requests on interface 3/1:

```
ServerIron(config)# interface e 3/1
ServerIron(config-if-3/1)# ip tcp syn-proxy out
```

Syntax: ip tcp syn-proxy in | out

When applied to inbound SYN requests, the SYN-Guard feature can be used with all ServerIron features, including TCS, FWLB, and SLB. However, when applied to outbound SYN requests, the SYN-Guard feature is the only process that can act on the packet.

NOTE: Do not configure the SYN-Defense feature simultaneously with the SYN-Guard feature.

Using the SYN-Guard Feature in High Availability Configurations

When used with FWLB and SLB, the SYN-Guard feature uses the high-availability method built into these components. When the SYN-Guard feature is not used with these components, you can provide redundancy with the following command:

```
ServerIron1(config)# server active-active-port e 3/12
```

Syntax: [no] server active-active-port ethernet <portnum> [<vlan-id>]

The <portnum> parameter is the first port MAC address where the peer ServerIron resides. This is the MAC address displayed as the "Boot Prom MAC" in the output of the **show chassis** command on the peer ServerIron. You must add a static MAC entry for this MAC address.

The <vlan-id> parameter specifies the VLAN you want to use for active-active synchronization traffic. Use this parameter if the port is a tagged member of multiple VLANs.

NOTE: The VLAN you specify must be used only for synchronization traffic. Do not specify a VLAN that also will carry data traffic.

To display the "Boot Prom MAC", enter the **show chassis** command on the peer ServerIron to display information such as the following:

```
ServerIron2# show chassis
power supply 1 ok
power supply 2 not present
power supply 1 to 2 from left to right
fan 1 ok
fan 2 ok
fan 3 ok
fan 4 ok
Current temperature : 34.5 C degrees
Warning level : 46 C degrees, shutdown level : 54 C degrees
Boot Prom MAC: 00e0.52c1.3700
```

On the other ServerIron, you would add a static MAC entry for the peer's "Boot Prom MAC" address. For example:

```
ServerIron1(config)# static-mac-address 00e0.52c1.3700 e 3/11
```

Syntax: static-mac-address <mac-addr> ethernet <portnum> [normal-priority | high-priority] [host-type | router-type]

SYN-Defense Support in SwitchBack (Direct Server Return)

ServerIron releases that do not support Layer 3 do not support the SYN-Defense feature in SwitchBack (Direct Server Return) configurations. This is because the ServerIron does not see the server's SYN ACK and as a result cannot complete the connection. The incomplete connection resides on the server as a pending connection, a condition the SYN-Defense feature is meant to eliminate.

TrafficWorks 8.0 enables you to use the SYN-Defense feature in a SwitchBack configuration. To do so, configure the server to use the ServerIron as its default gateway.

Transaction Rate Limiting

Transaction Rate Limiting (TRL) provides a way to monitor and limit traffic from any one IP address. When this feature is enabled, the ServerIron counts the number of bytes received from any one IP address over a specified interval. During this interval, if the number of bytes received from an individual IP address exceeds a specified threshold value, traffic from that IP address is held down and not processed for a specified number of minutes. You can use Transaction Rate Limiting to ensure that traffic from a single IP address does not monopolize resources on the ServerIron.

You apply Transaction Rate Limiting to individual interfaces on the ServerIron; only traffic on the specified interfaces is monitored. Transaction Rate Limiting can be applied to TCP, UDP, and ICMP traffic. For TCP and UDP traffic, you can apply Transaction Rate Limiting to up to four destination ports.

Transaction Rate Limiting, when used in conjunction with the SYN-Guard feature (where the ServerIron waits until a three-way handshake is completed with a connecting client before forwarding packets to the destination server), provides an additional defense against TCP flood attacks. The SYN-Guard feature shields the destination server from incomplete three-way handshakes, and the Transaction Rate Limiting feature causes TCP traffic from the attacker's IP address to be held down, and the attacker's IP address to be logged.

When traffic from an IP address is held down, a message is written to Syslog. In addition, you can display a list of the IP addresses whose traffic is being held, as well as statistics about that traffic.

To configure Transaction Rate Limiting on the ServerIron for TCP traffic, enter a command such as the following:

```
ServerIron(config)# ip tcp trans-rate monitor-interval 600 conn-rate 100 hold-down-
time 5
```

Syntax: ip tcp | udp | icmp trans-rate monitor-interval <interval> conn-rate <rate> hold-down-time <minutes>

The command above configures the ServerIron to monitor incoming TCP traffic. If any more than 100 TCP packets per second come from the same IP address over a 1 minute interval, then all TCP traffic from that IP address is held down for 5 minutes.

The **monitor-interval** <interval> specifies the amount of time used to measure incoming traffic. This parameter is specified in increments of 100ms. For example, to measure traffic over a 1 second interval, you would specify 10 for this parameter.

The **conn-rate** <rate> specifies a threshold for the number of bytes per second from any one IP address. Traffic exceeding this rate over the specified interval is subject to hold down.

The **hold-down-time** <minutes> specifies the number of minutes that traffic from an IP address that has sent packets at rate higher than the configured threshold is to be held down.

To apply Transaction Rate Limiting to a port on an interface, enter commands such as the following:

```
ServerIron(config)# interface ethernet 1/1
ServerIron(config-if-1/1)# ip tcp trans-rate 80
```

Syntax: ip tcp | udp trans-rate <ports>

Syntax: ip icmp trans-rate

The command above applies Transaction Rate Limiting to TCP traffic coming into port 80 on interface 1/1.

The <ports> parameter specifies one or more TCP or UDP ports to monitor. You can monitor up to 4 ports.

To display a list of IP addresses whose traffic has been held down, enter commands such as the following:

```
ServerIron# rconsole 2 1
ServerIron2/1 # show security holddown

source         destination    vers attempt start    last     HD time
192.168.2.30   Any tcp        ???? 0     000ab6ae   00000000   Y  9
192.168.2.40   Any tcp        ???? 0     000ab6ea   00000000   Y  9
```

Syntax: rconsole <slotnum> <cpunum>

Syntax: show security holddown

The following table lists the output from the **show security holddown** command:

Table 10.1: Output from the show security holddown command

This Field...	Displays...
source	Source IP address that is currently being held down
destination	TCP, UDP, or ICMP depending on the type of traffic sent by the client.
vers	Used by Foundry Technical Support.
attempt	Number of connection attempts made by the client during the current monitoring interval.
start	Time stamp representing the start of the monitoring interval.
last	Time stamp representing the last time the ServerIron received a connection request from the client.
HD	Whether the IP address is currently being held down. Y indicates that the address is being held down. N indicates that it isn't.

Table 10.1: Output from the show security holddown command (Continued)

This Field...	Displays...
time	Time remaining for this IP address to be held down, if the HD field contains Y.

To display statistics about traffic subject to Transaction Rate Limiting, enter commands such as the following:

```
ServerIron# rconsole 2 1
ServerIron2/1 # show security
```

Syntax: rconsole <slotnum> <cpunum>

Syntax: show security

Refusing Connections from a Specified IP Address

You can configure a ServerIron Chassis device to refuse connections from a specified IP address. Any new connection requests from the IP address are refused for a specified amount of time. This feature applies to all TCP, UDP, and ICMP traffic originating from the specified IP address.

To configure the ServerIron to refuse connections from 192.168.9.210 for 20 minutes, enter the following command:

```
ServerIron(config)# security hold-source-ip 192.168.9.210 20
```

Syntax: security hold-source-ip <ip-address> <minutes>

To display the IP addresses from which connections are currently being refused, enter commands such as the following:

```
ServerIron# rconsole 2 1
ServerIron2/1 # show security holddown

source         destination    vers attempt start    last     HD time
192.168.2.30   Any tcp        ???? 0     000ab6ae   00000000   Y  9
192.168.2.40   Any tcp        ???? 0     000ab6ea   00000000   Y  9
```

The IP addresses for which connections are being refused are displayed in the source column.

Logging Connection Information for DoS Attacks

You can configure the ServerIron to log information about the TCP connection rate and attack rate on the device.

To enable logging of TCP connection rate and attack rate, enter the following commands:

```
ServerIron(config)# ip tcp conn-rate conn-rate 10000 attack-rate 10000
ServerIron(config)# ip tcp conn-rate-change conn-rate 50 attack-rate 100
ServerIron(config)# server max-conn-trap 30
```

Syntax: ip tcp conn-rate conn-rate <rate> attack-rate <rate>

Syntax: ip tcp conn-rate-change conn-rate <percentage> attack-rate <percentage>

Syntax: server max-conn-trap <seconds>

The **conn-rate** <rate> parameter specifies a threshold for the number of global TCP connections per second that are expected on the ServerIron. A global TCP connection is defined as any packet that requires session processing. For example, 1 SLB, 1 TCS, and 1 SYN-Guard connection would equal 3 global TCP connections, since there are three different connections that require session processing.

NOTE: The ServerIron counts only the new connections that remain in effect at the end of the one second interval. If a connection is opened and terminated within the interval, the ServerIron does not include the connection in the total for the server.

The **attack-rate** <rate> parameter specifies a threshold for the number of TCP SYN attack packets per second that are expected on the ServerIron.

Syslog entries are generated under the following circumstances:

- If the connection rate or attack rate on the ServerIron reaches 80% of the configured threshold.
- If the connection rate or attack rate is still between 80% and 100% of the configured threshold 6 minutes after the last message.
- If the connection rate or attack rate exceeds 100% of the configured threshold.
- If the connection rate or attack rate exceeds 100% of the configured threshold, and has gone up by the configured rate change percentage.
- One minute after the last message indicating that the connection rate or attack rate still exceeds 100% of the configured threshold, and has gone up by the configured rate change percentage.
- Three minutes after the last message, if the connection rate or attack rate is still between 80% and 100% of the configured threshold, and has gone up by the configured rate change percentage.

The **server max-conn-trap** <seconds> command specifies the number of seconds that elapse between traps.

The **show server conn-rate** command shows the global TCP connection rate (per second) and TCP SYN attack rate (per second). This command reports global connection rate information for the ServerIron as well as for each real server. For example:

```
ServerIron# show server conn-rate
Avail. Sessions      =     524286  Total Sessions       =     524288
Total C->S Conn      =          0  Total S->C Conn      =          0
Total Reassign       =          0  Unsuccessful Conn    =          0
last conn rate       =          0  max conn rate        =          0
last TCP attack rate =          0  max TCP attack rate  =          0
SYN def RST          =          0  SYN flood            =          0
Server State - 1:enabled, 2:failed, 3:test, 4:suspect, 5:grace_dn, 6:active

Real Server     State   CurrConn    TotConn   LastRate   CurrRate    MaxRate
rs1                 3          0          0          0          0          0
```

Syntax: show server conn-rate

Chapter 11
Configuring CPU Protection

This chapter discusses the CPU Protection feature on the following devices:

- Devices with IronCore or JetCore modues running Enterprise software release 07.7.00 and later. See "CPU Protection for Enterprise Software Releases 07.7.00 and Later".
- NetIron IMR devices. See "Configuring Layer-2 CPU Protection on the NetIron IMR 640" on page 11-5

Refer to the section "Configuring Control Plane Security" on page 12-1 for CPU Protection on NetIron devices running software release 09.x.xx.

CPU Protection for Enterprise Software Releases 07.7.00 and Later

The ***CPU protection*** feature enhances the efficiency of a Foundry device's CPU and Content Addressable Memory (CAM).

Some denial of service attacks make use of spoofed IP addresses. If the device must create CAM entries for a large number of spoofed IP addresses over a short period of time, it requires excessive CAM utilization. Similarly, if an improperly configured host on the network sends out a large number of packets that are normally processed by the CPU (for example, DNS requests), it requires excessive CPU utilization.

The CPU protection feature allows you to configure the Foundry device to automatically take actions when thresholds related to high CPU or CAM usage are exceeded.

NOTE: The CPU protection feature is supported on the following devices, starting with software release 07.7.00:

- NetIron Chassis devices with IronCore or JetCore management modules
- BigIron devicees with IronCore or JetCore management modules
- FastIron II, FastIron II Plus, and FastIron III with M2 or higher modules
- FastIron 400, FastIron 800, and FastIron 1500 with JetCore modules
- FastIron 4802

The CPU protection feature is disabled by default.

Layer-2 CPU protection and Virtual Private LAN Service (VPLS) CPU Protection are also supported in release 02.1.00 for the NetIron IMR 640.

How the Feature Works

The CPU protection feature uses the concepts of ***normal mode*** and ***exhausted mode***. The device transitions from normal mode to exhausted mode when specified thresholds for ***conditions*** related to high CPU usage and

CAM usage are exceeded. When the device enters exhausted mode, ***actions*** can be taken to reduce the strain on system resources. You can define the conditions that cause the device to enter exhausted mode, the actions to take while the device is in exhausted mode, and the conditions that enable the device to go back to normal mode.

For example, you can specify that a CPU usage percentage of 90% is a condition that will cause the device to go from normal mode to exhausted mode. When the device enters exhausted mode, you can specify that the action to take is to forward unknown unicast traffic in hardware instead of sending it to the CPU. You can further specify that a CPU usage percentage of 80% will cause the device to go back to normal mode.

Conditions

You can define thresholds for the following conditions:

- CPU utilization percentage
- Layer 2, Layer 3, and Layer 4 CAM usage percentage

For each of these conditions, you can define two threshold values, a ***declaring watermark*** and a ***clearing watermark***. When the device is in normal mode, and a condition surpasses its declaring watermark, the device enters exhausted mode. When the device is in exhausted mode, and the condition drops below the clearing watermark, the device goes back into normal mode.

Actions

When the declaring watermark defined for a condition has been exceeded, the device enters exhausted mode. When the device is in exhausted mode, the following actions can be taken:

- ***Dynamic aging adjustment control*** – If dynamic aging adjustment control is specified as an action, when the system enters exhausted mode, the age limit value for CAM entries is dynamically changed to a smaller value, decreasing from 70 seconds to 35 seconds. When the system re-enters normal mode, the age limit value for CAM entries goes back to 70 seconds. Dynamic aging adjustment control is supported on both IronCore and JetCore devices.
- ***Unknown unicast flooding/dropping*** – You can configure the device to perform hardware flooding or dropping of unknown unicast packets when it enters exhausted mode. Packets with unknown unicast destination addresses can be either dropped or flooded by hardware to all ports in the VLAN. The unknown unicast flooding/dropping action is supported on JetCore devices only.
- ***Multicast/broadcast flooding/dropping*** – You can configure the system to drop or perform hardware flooding for multicast or broadcast packets, instead of sending them to the CPU. The multicast/broadcast flooding/dropping action is supported on JetCore devices only.

The hardware flooding actions are not applicable in every configuration, since under certain circumstances the device needs to send packets to the CPU for processing. For example, if a port on a Layer 3 Switch has an IP address configured, hardware flooding will not be enabled, so that ARP packets can be sent to the CPU.

Hardware flooding will not be applied on the following kinds of VLANs:

- Layer 2 control VLAN (VLAN ID 4094)
- Management VLAN
- Protocol VLAN
- Private VLAN

On a Layer 3 Switch, hardware flooding will not be enabled on a physical port that has a Layer 3 address configured. For virtual routing (VE) interfaces, packets are processed by the CPU by default, but hardware flooding can be enabled. See “Enabling Hardware Flooding on Virtual Routing Interfaces” on page 11-3 for more information.

Configuring CPU Protection

Configuring CPU protection consists of enabling CPU protection, specifying conditions that place the system into exhausted mode, and specifying actions to take when the system is in exhausted mode. The following sections instruct how to enable CPU protection and describe the default conditions and actions on the device, and how to modify the default conditions and actions.

Enabling CPU Protection

The CPU protection feature is disabled by default. To enable it, enter one or both of the following commands, depending on whether you want the device to protect the CPU, the CAM, or both.

- The following command enables the Foundry device to automatically take actions when thresholds related to high CAM usage are exceeded:

  ```
  BigIron(config)# cpupro-action hardware-flooding enable
  ```

  NOTE: Hardware flooding actions are supported on JetCore devices only.

  NOTE: To enable hardware flooding on virtual interfaces, see “Enabling Hardware Flooding on Virtual Routing Interfaces” on page 11-3.

- The following command enables the Foundry device to automatically take actions when thresholds related to high CPU usage are exceeded:

  ```
  BigIron(config)# cpupro-action quick-aging enable
  ```

Syntax: [no] cpupro-action hardware-flooding enable

Syntax: [no] cpupro-action quick-aging enable

Use the **no** form of the command to disable CPU protection.

Enabling Hardware Flooding on Virtual Routing Interfaces

By default, hardware flooding on virtual routing interfaces causes the device to copy packets to the CPU. You can optionally configure the device to allow hardware flooding without copying packets to the CPU.

To globally allow hardware flooding on virtual routing interfaces and disable the device from copying packets to the CPU, enter the following command:

```
BigIron(config)# cpupro-action hardware ve-not-to-cpu
```

Syntax: [no] cpupro-action hardware ve-not-to-cpu

To allow hardware flooding on a virtual routing interface for VLAN 10 and disable the device from copying packets to the CPU, enter the following command:

```
BigIron(config)# vlan 10
BigIron(config-vlan-10)# ve-flooding-not-to-cpu
```

Syntax: [no] ve-flooding-not-to-cpu

The setting for an individual VLAN overrides the global setting.

Default CPU Protection Conditions and Actions

By default, there are three CPU protection conditions defined on the device:

- Condition 1 is related to CPU usage. The declaring watermark is CPU usage at 90%, and the clearing watermark is CPU usage at 60%.
- Condition 2 is related to Layer 2, Layer 3, or Layer 4 CAM usage. The declaring watermark is Layer 2 CAM usage at 90%, and the clearing watermark is Layer 2 CAM usage at 60%.
- Condition 3 is a composite condition that is true If either Condition 1 or Condition 2 are true

Condition 3 is associated with the quick aging, unknown-unicast flooding, multicast flooding, and broadcast flooding actions. This means that when condition 3 is true, these actions will take place.

Modifying Default Conditions

You can modify the default CPU protection conditions listed above.

For example, to modify the default condition for CPU utilization percentage, enter a command such as the following:

```
BigIron(config)# cpupro-condition sys cpu declaring 95 clearing 60
```

Syntax: [no] cpupro-condition sys cpu declaring <percent> clearing <percent>

To modify the default condition for Layer 2, Layer 3, or Layer 4 CAM usage percentage, enter a command such as the following:

```
BigIron(config)# cpupro-condition sys cam declaring 95 clearing 60
```

Syntax: [no] cpupro-condition sys cam declaring <percent> clearing <percent>

For each condition, the declaring watermark percentage must be higher that the clearing watermark percentage.

Specifying Actions

For the hardware flooding actions, you can specify the number of CAM entries that can be allocated to each kind of traffic, as well as whether to flood or drop the traffic.

Allocating CAM Entries for Hardware Flooding

To accommodate the hardware flooding/dropping actions, the device allocates Layer 2 CAM entries to match broadcast traffic, multicast traffic, and unknown unicast traffic. These Layer 2 CAM entries exist in three sections of Layer 2 CAM space, one section for each type of traffic. By default, each of these sections consist of 256 CAM entries. You can optionally configure the number of CAM entries allocated for each type of traffic.

For example, to allocate 128 CAM entries for unknown-unicast flooding, enter the following command:

```
BigIron(config)# cpupro-action hardware unknown-unicast-flooding max-entries 128
```

Syntax: [no] cpupro-action hardware broadcast-flooding | multicast-flooding | unknown-unicast-flooding max-entries <entries>

Specifying Whether to Flood or Drop Traffic

When the unknown-unicast, broadcast, or multicast traffic actions are enabled, by default the device forwards the traffic in hardware, flooding it to all ports in the VLAN. You can optionally configure the device to drop the traffic instead.

For example, to cause unknown-unicast traffic to be dropped when the device enters exhausted mode, enter the following command:

```
BigIron(config)# cpupro-action hardware unknown-unicast-flooding mode drop
```

Syntax: [no] cpupro-action hardware broadcast-flooding | multicast-flooding | unknown-unicast-flooding mode flood | drop

Activating CPU protection for All VLANs

The commands **hardware-flooding**, **multicast-flooding**, and **broadcast-flooding**, which were available at the VLAN configuration level in previous Enterprise software releases, are no longer available in release Enterprise software release 07.8.01. Instead of these commands, use the global **cpupro-action** command to activate CPU protection for all VLANs configured on the device. This new method improves upon the previous hardware flooding method in that you do not need to reload the software when a VLAN is added or deleted. In addition, when configured, the new method applies globally to all VLANs, not just to individual VLANs.

For example, the following command enables hardware flooding for multicast traffic for all VLANs configured on the device:

```
BigIron(config)# cpupro-action hardware-flooding multicast-flooding on
```

To disable hardware flooding for multicast traffic for all VLANs configured on the device, enter the following command:

```
BigIron(config)# no cpupro-action hardware-flooding multicast-flooding on
```

Syntax: [no] cpupro-action hardware-flooding [multicast-flooding | broadcast-flooding | unknown-unicast-flooding] [on | off]

NOTE: Note that this new hardware flooding method works in conjunction with the CPU protection feature mentioned above. Foundry recommends that you do not use the new hardware flooding method at the same time that you use the CPU protection feature.

Specifying the Toggle Time Interval for Unknown Unicast Traffic

When hardware flooding is enabled for unknown unicast traffic, some unknown unicast packets are periodically sent to the CPU so that CAM entries can be created for individual destinations.

You can configure how often unknown unicast traffic is sent to the CPU by specifying the ***toggle time interval*** for unknown unicast traffic. The device alternates between flooding unknown unicast traffic and sending it to the CPU according to the specified toggle time interval.

For example, if you specify a toggle time interval of 10 seconds, the device will alternately flood unknown unicast traffic for 10 seconds, then send unknown unicast traffic to the CPU for 10 seconds.

To specify a toggle time of 10 seconds on the Foundry device, enter the following command:

```
BigIron(config)# cpupro-action unknown-unicast-toggle-time 10
```

Syntax: cpupro-action unknown-unicast-toggle-time <interval>

The <interval> can be from 1 – 60 seconds. The default is 5 seconds.

Refreshing the Layer 2 CAM Used for Hardware Flooding

In Enterprise software previous releases that supported CPU protection, you needed to disable and re-enable the feature whenever you added or removed a VLAN. Starting with release 07.8.01, you enter the **cpupro-action hardware-flooding refresh** command when you add or remove a VLAN. This command refreshes the Layer 2 CAM used for hardware flooding. You no longer need to disable and re-enable CPU protection when you add or remove a VLAN.

For example, the following commands add a port to VLAN 111 and refresh the Layer 2 CAM used for hardware flooding:

```
BigIron(config)# vlan 111
BigIron(config-vlan-111)# untagged e 2/20
BigIron(config-vlan-111)# exit

BigIron(config)# cpupro-action hardware-flooding refresh
```

Syntax: [no] cpupro-action hardware-flooding refresh

Configuring Layer-2 CPU Protection on the NetIron IMR 640

NOTE: This feature is available in release 02.1.00 for the NetIron IMR 640.

This feature protects the line-card's CPU from being overwhelmed by excessive L2 packets that would require the CPU's attention, including known unicast, multicast packets and packets requiring source-MAC learning. Once this feature is enabled, all L2 multicast traffic will be hardware flooded. Furthermore, when the CPU is too busy, this feature will hardware flood unknown unicast traffic, as well as reduce the rate of source-MAC learning traffic to the line-card's CPU, so that the line-card's CPU will have enough resources to handle other types of packets.

Configuring Layer-2 CPU Protection

Layer-2 CPU protection is enabled per VLAN. To enable Layer-2 CPU protection on a VLAN, enter the following command:

```
NetIron IMR640 Router(config)# vlan 24
NetIron IMR640 Router(config-vlan-24)# vlan-cpu-protection
```

Syntax: [no] vlan-cpu-protection

By default, a maximum of 8 VLANs can have this feature enabled. If you want to increase the number, you must use the following command:

```
NetIron IMR640 Router(config)# system-max hw-flooding 40
```

Syntax: [no] system-max hw-flooding <number>

The <number> variable is the number of VLANs that you want to support with Layer-2 CPU protection.

NOTE: You must reload your system for the **system-max** command to take effect.

Configuring VPLS CPU Protection on the NetIron IMR 640

NOTE: This feature is available in release 02.1.00 for the NetIron IMR 640.

This feature protects the line-card's CPU from being overwhelmed by excessive VPLS packets that would require the CPU's attention, including known unicast, multicast packets and packets requiring source-MAC learning. Once this feature is enabled, all VPLS multicast traffic will be hardware flooded. Furthermore, when the CPU is too busy, this feature will hardware flood unknown unicast traffic, as well as reduce the rate of source-MAC learning traffic to the line-card's CPU, so that the line-card's CPU will have enough resources to handle other types of packets.

Configuring Adequate CAM Resources

When using VPLS CPU protection, you must have adequate CAM resources available. Each end-point and each uplink port requires a single CAM entry. Also, if an end-point is a trunk port, one entry is required for each port in the trunk. To determine the number of entries required on your system, add the number VPLS end-points, ports within a trunk port used as an end-point, and uplink ports. Use this number with the **system-max hw-flooding** command to provide adequate CAM resources.

By default, 8 CAM entries are available. To configure a larger number, you must use the following command:

```
NetIron IMR640 Router(config)# system-max hw-flooding 40
```

Syntax: [no] system-max hw-flooding <number>

The <number> variable is the number of CAM entries that you want to make available.

NOTE: You must reload your system for the **system-max** command to take effect.

Configuring VPLS CPU Protection

VPLS CPU protection can be configured in either of the following two ways:

- Globally – This enables VPLS CPU protection to effect all VPLS instances on the router.
- Per-VPLS – This enables VPLS CPU protection on one or more specified VPLS instances.

Configuring VPLS CPU Protection Globally

VPLS CPU protection can be enabled for all VPLS instances on a router. To enable VPLS CPU protection on all VPLS instances, enter the following command:

```
NetIron IMR640 Router(config)# router mpls
NetIron IMR640 Router(config-mpls) vpls-cpu-protection
```

Syntax: [no] vpls-cpu-protection

Configuring VPLS CPU Protection Per VPLS

VPLS CPU protection can be enabled per VPLS instance. To enable VPLS CPU protection on a specified VPLS instance, enter the following command:

```
NetIron IMR640 Router(config)# router mpls
NetIron IMR640 Router(config-mpls) vpls test 1
NetIron IMR640 Router(config-mpls-vpls-test)# cpu-protection
```

Syntax: [no] cpu-protection

Displaying CPU Protection Information

You can display information about the conditions and actions configured on the device.

Displaying Condition Information

To display the CPU protection conditions configured on the device, enter the following command:

```
BigIron# show l2-cpupro conditions

Condition Configuration:
01: 00000001 Atomic cpu dwm: 90, cwm: 60
02: 00000002 Atomic cam layer2 dwm: 90, cwm: 60
03: 4 Composite or(1,2)

System Condition Monitoring: Normal.
CPU Condition: Normal.
        DWM:90 CWM:60 CUR:1 CNT:0
Layer 2 CAM Condition: Normal.
        DWM:90 CWM:60 CUR:0 CNT:0
```

Syntax: show l2-cpupro conditions

Table 11.1 lists the output of the **show l2-cpupro condition** command.

Table 11.1: Output of the show l2-cpupro condition command

This field	Displays
Condition Configuration:	The conditions configured on the device. These include the three pre-configured conditions, as well as any user-configured conditions.
Condition Monitoring:	Whether any of the conditions has surpassed its declaring watermark.
CPU Condition:	Whether the condition defined for CPU usage percentage has surpassed its declaring watermark, as well as the condition’s declaring watermark, clearing watermark, and the number of times the declaring watermark has been exceeded.
Layer 2 CAM Condition:	Whether the condition defined for Layer 2 CAM usage percentage has surpassed its declaring watermark, as well as the condition’s declaring watermark, clearing watermark, and the number of times the declaring watermark has been exceeded.

Displaying Action Information

To display information about actions configured on the device, enter the following command:

```
BigIron# show l2-cpupro actions

Action Configuration:
1: action(03) q-aging(Ena) Deactivated
2: action(03) hw-flood(Ena) Deactivated

Action Execution:
Quick Aging: enabled
        Not in Quick Aging mode
        Normal age limit: 126
        Quick age limit: 63

HW Flooding:
unknown-unicast: installed not activated
        max entries: 256, current entries: 5
        action mode:flooding
multicast: installed not activated
        max entries: 256, current entries: 5
        action mode:flooding
broadcast: installed not activated
        max entries: 256, current entries: 5
        action mode:flooding
```

Syntax: show l2-cpupro actions

Table 11.2 lists the output of the **show l2-cpupro actions** command.

Table 11.2: Output of the show l2-cpupro actions command

This field	Displays
Action Configuration:	The actions configured on the device.
Action Execution:	Which actions have been enabled.
Quick Aging:	Whether the quick aging action has been enabled for a condition.
Not in Quick Aging mode	Whether the device is in exhausted mode and the quick aging action is in effect.
Normal age limit:	The age limit for CAM entries when the device is in normal mode.
Quick age limit:	The age limit for CAM entries when the device is in exhausted mode and the quick aging action is in effect.
HW Flooding:	Information about the three kinds of hardware-flooding actions.

Table 11.2: Output of the show l2-cpupro actions command

This field	Displays
unknown-unicast:	Whether CAM entries for the unknown-unicast action have been installed and whether the unknown-unicast action has been enabled: max entries: – The maximum number of Layer 2 CAM entries that can be allocated for hardware flooding of unknown-unicast traffic. current entries: – The number of Layer 2 CAM entries currently in use for hardware flooding of unknown-unicast traffic. action mode: – Whether the device floods or drops unknown-unicast traffic when this action takes place.
broadcast:	Whether CAM entries for the broadcast action have been installed and whether the broadcast action has been enabled: max entries: – The maximum number of Layer 2 CAM entries that can be allocated for hardware flooding of broadcast traffic. current entries: – The number of Layer 2 CAM entries currently in use for hardware flooding of broadcast traffic. action mode: – Whether the device floods or drops broadcast traffic when this action takes place.
multicast:	Whether CAM entries for the multicast action have been installed and whether the multicast action has been enabled: max entries: – The maximum number of Layer 2 CAM entries that can be allocated for hardware flooding of multicast traffic. current entries: – The number of Layer 2 CAM entries currently in use for hardware flooding of multicast traffic. action mode: – Whether the device floods or drops multicast traffic when this action takes place.

Displaying VPLS CPU Protection Configuration Status

To view the VPLS CPU protection configuration status for a specified VPLS instance, use the show **mpls vpls id** command as shown:

```
NetIron IMR640 Router(config)# show mpls vpls id 1
VPLS test1, Id 1, Max mac entries: 2048
 Total vlans: 1, Tagged ports: 1 (0 Up), Untagged ports 1 (1 Up)
  Vlan 2
   Tagged: ethe 5/4
   Untagged: ethe 2/2
 Total VC labels allocated: 32 (983040-983071)
 Total VPLS peers: 1 (0 Operational)
 Peer address: 1.1.1.1, State: Wait for remote VC label from Peer
  Tnnl: tnl0(3), LDP session: Up, Local VC lbl: 983040, Remote VC lbl: N/A
 CPU-Protection: ON,  MVID: 0x000,  VPLS FID: 0x00000205
```

The CPU protection status is shown in **bold**. It can be either on or off.

Chapter 12
Configuring Control Plane Security

Overview

NOTE: This feature is supported on NetIron devices with JetCore modules running Service Provider software release 09.1.00 or higher.

The ***Control Plane Security*** feature prevents the Foundry device's CPU from being overloaded when the device receives a large amount of unknown Layer 2 traffic, or when the device's Layer 2 Content Addressable Memory (CAM) is full.

CAM is a component of Foundry modules that facilitates hardware-based forwarding. As packets flow through the Foundry device from a given source to a given destination, the Foundry device's CPU records forwarding information about the flow in CAM entries. CAM entries contain Layer 2, Layer 3, or Layer 4 next-hop information. Each type of CAM entry has its own format; for example, Layer 2 CAM entries contain destination MAC address, 802.1p (priority), and VLAN information, while Layer 3 CAM entries contain destination IP information. Incoming packets are matched against entries in the CAM. If the packet matches a CAM entry, it is forwarded by hardware, without the aid of the CPU, which speeds up forwarding time and reduces the strain on the CPU.

Unknown traffic (that is, traffic for which there is no CAM entry) is processed by the CPU (software-based forwarding), which creates CAM entries for the traffic. When the Foundry device receives a large amount of unknown traffic, the CPU can expend much of its resources doing software-based forwarding. When this happens, CPU usage on the device can reach 100 percent. Additionally, if the CAM fills to capacity, unknown traffic is forwarded by the CPU until existing CAM entries are aged out. When packets are forwarded by the CPU instead of by hardware, at a high traffic rate, it can also cause CPU usage to reach 100 percent. If the CPU is used at 100 percent capacity for an extended period of time, packets may be dropped, and it may be difficult to ping or Telnet to the Foundry device.

The Control Plane Security feature reduces the likelihood of CPU usage reaching 100 percent by forwarding unknown Layer 2 traffic destined for specified VLANs in hardware, bypassing the CPU. When the feature is configured on a VLAN, a Layer 2 CAM entry is created that causes unknown traffic destined for the VLAN to be broadcast by hardware to all ports in the VLAN. For a VLAN that has this feature enabled, two kinds of CAM entries are possible: CAM entries for individual destinations, and a “match-all” CAM entry that applies to all packets destined for the VLAN. Any traffic for that VLAN that doesn't match a CAM entry for an individual destination will match the match-all CAM entry. Traffic that matches the match-all CAM entry is hardware-flooded to all ports in the VLAN, reducing the burden on the CPU.

To allow CAM entries to be made for individual destinations in VLANs where Control Plane Security is enabled, the CPU periodically processes unknown Layer 2 unicast, broadcast, and multicast traffic.

To learn the source addresses of the devices connected to it, the Foundry device sends packets with unknown source addresses to the CPU, which records this information in Layer 2 CAM entries. When the Control Plane

Security feature is configured on at least one VLAN, if the Foundry device detects that Layer 2 CAM is full for the IPC that handles the VLAN, then source address learning, CAM programming, and station movement for ports on that IPC is controlled by the match-all CAM entry as well. Source address learning packets for the VLAN are sent to the CPU only periodically until the IPC's Layer 2 CAM is no longer full.

By reducing the amount of source address learning traffic sent to the CPU after Layer 2 CAM is full, the likelihood of CPU usage reaching 100 percent due to continuous processing of source address learning traffic when Layer 2 CAM is full is diminished.

NOTE: When the amount of source address learning traffic sent to CPU is reduced, the MAC address learning rate is reduced. When the Foundry device's MAC address table is large (greater than 250,000 entries) and Layer 2 CAM on an IPC is full, since Layer 2 traffic is mostly hardware-forwarded instead of being sent to CPU for software-based forwarding and MAC learning, the MAC entries in the software MAC address table may not be refreshed in time and can be aged out, even though traffic is being received at an interface.

To prevent this from happening, you can either reduce the CPU protection control-timer value (see "Configuring the Timer for Control Plane Security" on page 12-3) so that packets can come to CPU more often to refresh the entries in the MAC address table, or you can increase the default MAC aging time.

When Layer 2 CAM is full, and there is no additional space to program a CAM entry for a specific MAC address, traffic for that specific MAC address is hardware flooded to all ports of the VLAN instead of forwarded out an individual port, even though the MAC address is already learned.

The Control Plane Security feature is supported for unknown unicast, multicast, or broadcast traffic, unless an IP address is configured on an untagged port, or the VLAN has a Virtual Ethernet (VE) interface, in which case the feature is not supported for multicast or broadcast traffic. For unknown unicast, multicast, or broadcast traffic received on a trunk group, this feature is supported only on one port of the trunk.

NOTE: This feature is not supported on 10 Gigabit Ethernet interfaces.

Configuring Control Plane Security

Configuring the control plane security feature consists of the following tasks:

- Enabling hardware flooding for a VLAN
- Configuring how often the CPU can process unknown Layer 2 unicast, broadcast, and multicast traffic (or source address learning packets when Layer 2 CAM is full) for VLANs where hardware flooding is enabled (optional)
- Specifying the number of VLANs to which Control Plane Security can apply (optional)

Enabling Hardware Flooding for a VLAN

When you enable hardware flooding for a VLAN, unknown unicast packets are flooded by hardware to all ports in the VLAN. A Layer 2 entry is made in the Foundry device's CAM that matches all packets for that VLAN. Any traffic for the VLAN that doesn't match a CAM entry for an individual destination matches this CAM entry.

To enable hardware flooding for a VLAN, enter commands such as the following:

```
BigIron(config)# vlan 222
BigIron(config-vlan-222)# hardware-flooding
```

Syntax: [no] hardware-flooding

The **hardware-flooding** command takes effect immediately; you do not have to reboot the device.

You can configure the **hardware-flooding** command for an individual VLAN or for VLAN groups. You cannot configure the **hardware-flooding** command on a VLAN if any of the following are true:

- The VLAN is a protocol VLAN
- The VLAN is a private VLAN

- The VLAN is a management VLAN
- Multicast hardware flooding is enabled on the VLAN
- The **route-only** command is configured globally
- An IP subnet VLAN is configured
- The VLAN does not contain at least one port

If Layer 2 switching is disabled on an interface (with the **route-only** command), traffic received on the interface is not broadcast to all VLAN ports by hardware, even though the **hardware-flooding** command is configured.

Configuring the Timer for Control Plane Security

When hardware flooding is enabled for a VLAN, unknown Layer 2 unicast, broadcast, and multicast traffic destined for the VLAN is periodically sent to the CPU so that CAM entries can be created for individual destinations. You can configure how often this traffic is sent to the CPU by specifying a value for the CPU protection control timer.

The CPU protection control timer controls how frequently the CPU processes unknown Layer 2 unicast, broadcast, and multicast traffic matching the match-all CAM entries on different IPCs belonging to different VLANs. The default is 1000 milliseconds (1 second), which means that every second, the CPU can process unknown Layer 2 unicast, broadcast, and multicast traffic (and source address learning packets when Layer 2 CAM is full) for a period of about 100ms. If you change the timer to 500ms, then the CPU can process unknown packets for 100ms, during a 500ms time period. The 100ms processing period is not configurable.

To set the CPU protection control timer to 800 milliseconds, enter the following commands:

```
BigIron(config)# cpu-protection
BigIron(config-cpu-protection)# control-timer 8
```

Syntax: cpu-protection

Syntax: [no] control-timer <time>

The <time> parameter specifies an amount of time in units of 100 milliseconds. You can specify from 2 – 100. The default is 10 (1000 milliseconds, or 1 second).

Specifying the Number of VLANs to Which Control Plane Security Can Apply

By default, the Control Plane Security feature can be enabled on up to 1024 VLANs (512 on the NetIron 4802). You can change this maximum to between 0 and the maximum number of VLANs supported on the device.

To set the maximum number of VLANs where the Control Plane Security feature can be enabled to 2048, enter the following commands:

```
BigIron(config)# cpu-protection
BigIron(config-cpu-protection)# max-vlans 2048
```

Syntax: [no] max-vlans <number>

Note that increasing this value reduces the amount of Layer 2 CAM space available for individual destinations.

You must save the configuration with the **write memory** command and reboot the Foundry device in order for this new setting to take effect.

Configuring Drastic CPU Protection Scheme

NOTE: The drastic CPU Protection Scheme is available on Service Provider software release 09.2.00 and later.

The drastic CPU protection scheme provides the following:

- It ensures that MRP control packets are always processed by the management module.
- When CPU usage is above a trigger point or threshold, the drastic CPU protection scheme is initiated and does the following:

1. Temporarily discards unknown broadcast or multicast packets.
2. Disables SA learning.
3. Disables unknown unicast broadcasting.

   Once the traffic falls below the low trigger point or threshold, the drastic CPU protection scheme is disabled and hardware flooding scheme resumes.

The drastic CPU protection scheme is enabled on VLANs that have the hardware flooding option enabled. During normal traffic, hardware flooding is used to protect the CPU. Once CPU usage goes above a certain percentage, drastic CPU protection scheme can then be implemented.

To enable drastic CPU protection scheme, use the CPU Protection Watermarking option. Enter a command such as the following:

```
NetIron(config)# cpu-protection
NetIron(config-cpu-protection)# cpu-usage high-water-mark 70 low-water-mark 40
```

Syntax: [no] cpu-usage high-water-mark <percentage> low-water-mark <percentage>

When the CPU usage goes above the defined highwater mark, the CPU puts the drastic CPU protection scheme into effect. Once CPU usage goes below the low watermark, the CPU ends the drastic CPU protection scheme.

EXAMPLE:

The following configuration assumes VLAN 20 has already been defined and hardware flooding is enabled on that VLAN. During the normal traffic, hardware flooding will be used as long as the CPU usage rate does not go over 70%. Once the CPU usage rate exceeds 70% of the available CPU, the drastic CPU protection scheme is initiated. to lower the CPU usage rate.

Then if CPU usage rate drops below 40%, the drastic CPU protection scheme will be stopped and hardware flooding resumes.

```
NetIron(config)# vlan 20
NetIron(config-vlan)# hardware-flooding
NetIron(config)# exit

NetIron(config)# cpu-protection
NetIron(config-cpu-protection)# cpu-usage high-water-mark 70 low-water-mark 40
NetIron(config-cpu-protection)# exit
```

To display the configured watermarking option enter a command such as the following:

```
NetIron#show cpu-protection

Number of vlans configured with hardware-flooding: 2
Timer for cpu forward: 1000ms

CPU usage high watermark: 60, Hit count: 0
CPU usage low watermark: 20, Hit count: 0
CPU high watermark protection state currently: OFF
VLAN 20: l3_support ON
VLAN 30: l3_support OFF
```

Syntax: show cpu-protection

The show **cpu-protection** command shows some CPU protection configuration information, as well as statistics for the number of times drastic protection scheme was invoked. It also displays a list of all VLANs with hardware-flooding configured.

Chapter 13
Configuring Reverse Path Forwarding

This chapter about Reverse Path Forwarding (RPF) covers the following topics:

- Configuring Unicast RPF
- Configuring RPF in the software release 02.2.00 and later for the BigIron MG8 and NetIron 40G
- Configuring RPF in the software release 02.1.00 and later for the NetIron IMR 640
- Displaying unicast RPF information
- Displaying RPF Statistics for the NetIron IMR 640

Overview of Unicast RPF

Foundry devices support unicast RPF. It can be used as a defense against Denial of Service (DoS) attacks in which an attacker attempts to flood a network with packets that have spoofed or falsified source IP addresses.

When unicast RPF is enabled on an interface, the Foundry device examines the source address of each incoming packet and verifies that this interface is the correct one to receive packets with that source address. If this is not the correct interface to receive the packet, then the packet is dropped.

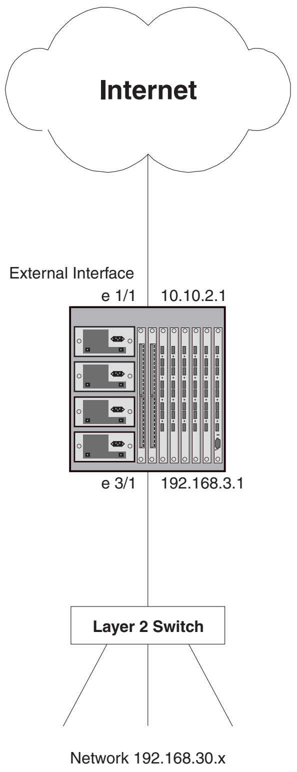
For example, in one type of DoS attack an attacker may attempt to send a large amount of traffic into a network through an external interface, using spoofed addresses of the internal network as the source of the traffic. When the unicast RPF feature is enabled, the Foundry device recognizes that traffic coming in on an external interface should not have source addresses belonging to the internal network, and the Foundry device consequently drops the traffic with the spoofed source addresses.

When you configure unicast RPF on an interface, you specify which interfaces are ***external***. External interfaces are those that can receive traffic from external networks, but not from the internal network. When an interface is specified as an external interface for unicast RPF, the Foundry device creates CAM entries that correspond to the internally learned routes in the device's routing table. The unicast RPF CAM entries deny packets with source IP addresses that correspond to the internally learned routes. All other traffic is permitted.

To accommodate routing table changes, the internally learned routes are synchronized with the unicast RPF CAM entries every 1 minute.

Figure 12.1 shows an illustration of a network that uses unicast RPF.

Figure 13.1 Unicast RPF configuration

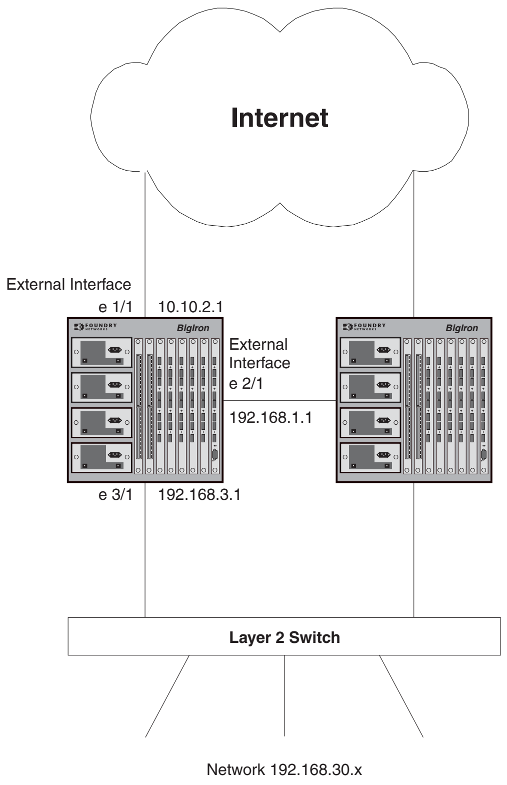


In this configuration, interface e 1/1 is specified as an external interface for unicast RPF. CAM entries are created for that interface that deny incoming packets with source addresses from the 192.168.9.x network.

The Foundry device's loopback interface/network is also considered an internally learned route. In the example, Incoming packets on interface e 1/1 that have a source address corresponding to the Foundry device's loopback interface/network are dropped.

For interfaces that can receive packets from the internal network as well as from external sources, you identify the interface as an external interface; this prevents the Foundry device from creating RPF CAM entries for routes learned on the interface. For example, in the configuration in Figure 12.2, interface 2/1 can receive packets from the Internet as well as from the internal network.

Figure 13.2 Unicast RPF configuration with an interface identified as external



In this example, interface 2/1 as identified as an external interface. When the Foundry device compiles the list of internally learned routes for unicast RPF, it does not include the routes learned on interface 2/1. Note that identifying an external interface in this way (using the **ip verify unicast external-interface** command, rather than the **ip verify unicast reverse-path external** command) does not enable unicast RPF for incoming packets on the interface. In the example, unicast RPF CAM entries are not created for routes learned on interface 2/1, and incoming traffic on interface 2/1 is not permitted or denied using unicast RPF CAM entries.

NOTE: Unicast RPF is supported on Foundry devices running Enterprise software release 07.8.00 or higher. This feature is supported on JetCore devices only. It is not supported on 10 Gigabit Ethernet interfaces.

If the external interface is a POS interface, then routes from internal loopback interface networks, or routes whose destination is in an IP prefix list, are not subject to unicast RPF.

Configuring Unicast RPF

To configure unicast RPF, you enable it on an interface and specify the interface as external. Note that unicast RPF applies to incoming traffic on an interface where it is configured.

For example, the following commands enable unicast RPF on interface e 1/1, and identify it as an external interface:

```
BigIron# interface e 1/1
BigIron(config-if-e100-1/1)# ip verify unicast reverse-path external
```

When unicast RPF is enabled on an external interface, packets with source IP addresses that correspond to the internally learned routes are denied. All other traffic is permitted.

Syntax: [no] ip verify unicast reverse-path external

To identify an interface as an external interface, without enabling unicast RPF on the interface, enter commands such as the following:

```
BigIron# interface e 2/1
BigIron(config-if-e100-2/1)# ip verify unicast external-interface
```

When an interface is identified as an external interface with this command, it prevents the Foundry device from creating RPF CAM entries for routes learned on the interface. Unicast RPF is not performed for incoming packets on the interface.

Syntax: [no] ip verify unicast external-interface

Specifying a Prefix List for Unicast RPF

When unicast RPF is enabled on an external interface, the Foundry device compiles a list of the internally learned routes in the device's routing table and creates unicast RPF CAM entries that deny packets with source IP addresses corresponding to these routes.

In addition, you can create an IP prefix list containing a list of routes and then specify the IP prefix list as part of the unicast RPF configuration. When you do this, the Foundry device creates unicast RPF CAM entries that deny packets with source IP addresses corresponding to the routes in the IP prefix list, in addition to the internally learned routes. Using an IP prefix list in this way allows you to configure the device to deny packets from networks other than internal ones.

For example, the following commands create an IP prefix list called "martians" consisting of three routes:

```
BigIron(config)# ip prefix-list martians seq 5 deny 0.0.0.0/8 le 32
BigIron(config)# ip prefix-list martians seq 10 deny 10.0.0.0/8 le 32
BigIron(config)# ip prefix-list martians seq 15 deny 127.0.0.0/8 le 32
```

NOTE: For information on creating IP prefix lists, see the *Foundry Enterprise Configuration and Management Guide*.

The following commands specify the "martians" IP prefix list as part of the unicast RPF configuration for interface e 1/1:

```
BigIron# interface e 1/1
BigIron(config-if-1/1)# ip verify unicast reverse-path external prefix-list martians
```

Syntax: [no] ip verify unicast reverse-path external prefix-list <name>

This command causes the device to drop packets incoming on interface e 1/1 that have source addresses corresponding to the routes in the "martians" prefix list (as well as packets with source addresses corresponding to internally learned routes, as well as the device's loopback address/network).

Displaying Unicast RPF Information

To display information about the CAM entries created for unicast RPF, enter the following command:

```
BigIron# show ip rpf
Total number of RPF route entries 4
        Destination     NetMask          Port    VLAN ID    Deny
        40.0.0.0        255.255.255.0    3/9        1      32778
        30.0.0.0        255.255.255.0    3/9        1      32779
        192.168.135.0   255.255.255.0    v2         2      32769
         65.0.0.0         255.255.255.252 3/9          1       32770
```

Syntax: show ip rpf [<ipaddr> | <ipaddr>/<mask-length> | <portnum> | <ve>]

Table 12.1 lists the information displayed in the output of the **show ip rpf** command.

Table 13.1: Output of the show ip rpf command

This Field...	Displays...
Total number of RPF route entries	The number of CAM entries that have been created for unicast RPF.
Destination	The address of the route.
NetMask	The subnet mask for the route.
Port	The port on which the route was learned.
VLAN ID	The VLAN ID of the port.
Deny	The index number of the CAM entry used for denying packets matching this route.

Clearing Unicast RPF CAM Entries

To clear the CAM entries created for unicast RPF, enter the following command:

```
BigIron# clear ip rpf
```

Syntax: clear ip rpf

Configuring RPF

NOTE: RPF is available in release 02.2.00 and later for the BigIron MG8 and NetIron 40G, and release 02.1.00 and later for the NetIron IMR 640.

A number of common types of denial-of-service (DoS) attacks, including Smurf and Tribe Flood Network (TFN), can take advantage of forged or rapidly changing source IP addressed to allow attackers to thwart efforts to locate or filter the attacks. RPF is designed to prevent such a malicious user from spoofing a source IP address by checking that the source address specified for a packet is received from a network that the BigIron MG8 or NetIron 40G have access to. Packets with invalid source addresses aren't forwarded.

Configuring RPF on the BigIron MG8 and NetIron 40G

To configure a BigIron MG8 or NetIron 40G for RPF, perform the following steps:

- Configure the global command: reverse-path-check.
- Configure a CAM partition for RPF.
- Enable RPF mode individually on ports that you want it to run.

Configuring the Global RPF Command

Before you can enable RPF to operate on a BigIron MG8 or NetIron 40G, you must first configure RPF globally as shown:

```
BigIron MG8(config)# reverse-path-check
```

Syntax: [no] reverse-path-check

You must reload the system for this command to take effect.

Configuring a CAM Partition for RPF

By default there is no CAM reserved to run RPF. To run RPF, you must reconfigure the CAM partitioning by block to contain one or more blocks for RPF. The default CAM block partitions are described in "Changing CAM

Partitions” in the *Foundry Diagnostic Guide*. In Terathon software release 02.2.00 and later for the BigIron MG8 and NetIron 40G, a new parameter is added, allowing you to reserve one or more CAM partition blocks for RPF.

In the following example, the CAM partition blocks are being changed from the default by eliminating the blocks for IPv6 from the default and adding 2 blocks for RPF. To create this CAM partition configuration, you can use the following command:

```
BigIron (config)# cam-partition block session-mac 2 ip-mac 2 out-session 2 ipv6 0
ipv6-session 0 rpf 2
```

Syntax: cam-partition block session-mac <blocks_allocated> ip-mac <blocks_allocated> out-session <blocks_allocated> ipv6 <blocks_allocated> ipv6-session <blocks_allocated> rpf <blocks_allocated>

<blocks_allocated> specifies the number of blocks allocated to the specified allocation parameter. A total of 4 blocks are available for 9 meg interface modules (called LV or low value) and 8 blocks for 19 meg interface (called HV or high value) modules. Use zero (0) for parameters that you do not want to allocate a CAM partition block for.

Enable RPF on Individual Ports

After RPF has been configured globally for a BigIron MG8 or NetIron 40G, it must be configured on every interface that you want it to operate. It can be configured at each interface for either "strict" mode to check if there is a valid route to the specific port being configured or "loose" mode which only checks for a valid route to the router. To configure RPF on a port, use as command such as the following:

```
BigIron MG8(config)# interface ethernet 3/1
BigIron MG8(config-if-e1000-3/1)# rpf-mode strict
```

Syntax: rpf-mode [loose | strict]

The **loose** parameter directs RPF to deny packets with source addresses from entering this port it there is no valid path to the router as determined by the routing protocols.

The **strict** parameter directs RPF to deny packets with source addresses from entering this port it there is no valid path to the port as determined by the routing protocols.

Configuring RPF on the NetIron IMR 640

To configure NetIron IMR 640 for RPF, you must perform the following steps:

- Considerations for configuring RPF with ECMP routes
- RPF Support for IP over MPLS routes
- Set your CAM mode to be compatible with RPF
- Configure the global command: **reverse-path-check**
- Enable RPF mode individually on ports that you want it to run.
- Suppressing RPF for Packets with Specified Address Prefixes

Considerations for Configuring RPF with ECMP Routes

For a source IP address matching an ECMP route, RPF will permit the packet if it arrives on any of the next-hop interfaces for that route. For example, if there are two best next-hops for a network route 11.11.11.0/24, one pointing to 10.10.10.1 (Gigabit Ethernet 7/1) and the other to 10.10.30.1 (Gigabit Ethernet 7/12), then incoming packets with source address matching 11.11.11.0/24 will be permitted on either Gigabit Ethernet 7/1 or Gigabit Ethernet 7/12.

A disadvantage of this configuration is that if some other route shares any of these next-hops, the packets with a source IP address matching that route will also be permitted from any of the interfaces associated with those next hops. For example, say 12.12.12.0/24 has the next-hop 10.10.10.1, then packets from 12.12.12.0/24 will also be be permitted on either Gigabit Ethernet 7/1 or Gigabit Ethernet 7/12.

RPF Support for IP over MPLS routes

For IPv4 routes over MPLS tunnels, the physical interface for an outgoing tunnel on which a route is assigned may not be the same as the one from which we receive packets. Consequently, only RPF loose mode is supported on MPLS uplinks.

Setting CAM mode to RPF Compatibility

RPF requires that CAM be configured with the following parameters:

- CAM mode must be set to static.
- The RPF feature must operate with at least 512K IPv4 CAM entries per PPCR or less. Consequently, the CAM profile can be any of the possible profiles except the IPv4 profile. For a description of the available CAM profiles, see the chapter on CAM partitioning in the *Foundry Diagnostic Guide*.

NOTE: Since the RPF feature requires that the entire IP route table is available in hardware, the feature must work in conjunction with Foundry Direct Routing (FDR). FDR is the default mode of operation for the NetIron IMR 640. For more information about enabling and disabling FDR, see the FDR chapter in the *Foundry Diagnostic Guide*.

Configuring the Global RPF Command

Before you can enable RPF to operate on a NetIron IMR 640 you must first configure RPF globally as shown:

```
NetIron IMR640 Router(config)# reverse-path-check
```

Syntax: reverse-path-check

NOTE: If you attempt to enable the global RPF command on a system with incompatible CAM settings, the command will be rejected and you will receive a console message describing this.

Enable RPF on Individual Ports

After RPF has been configured globally for a NetIron IMR 640, it must be configured on every interface that you want it to operate. The RPF feature can only be configured on physical ethernet interfaces. There are two modes "strict" and "loose" that can be configured to enforce RPF on IP addresses for packets arriving on a given interface, as described in the following:

- In **loose** mode, RPF will permit a packet as long as the source address matches a known route entry in the routing table. It will drop a packet if it doesn't match a route entry. Please note that if a default route is present, loose mode will permit all traffic.
- In **strict** mode, RPF requires that a packet matches a known route entry as described in loose mode and also that it arrives at the interface as described in the router table's next hop information. It will drop a packet that doesn't match both of these criteria.

To configure RPF on a port, use as command such as the following:

```
NetIron IMR640 Router(config)# interface ethernet 3/1
NetIron IMR640 Router(config-if-e1000-3/1)# rpf-mode strict log
```

Syntax: [no] rpf-mode [loose | strict] [log]

There are two modes in which you can enforce RPF on IP sources address for packets that arrive on a configured interface: **loose** mode and **strict** mode as described previously.

NOTE: If a default route is present on the router, loose mode will permit all traffic.

The **log** parameter directs RPF to log packets that fail the RPF test. Enabling RPF logging may lead to high CPU utilization on the interface module because packets that fail the RPF check test are dropped in software. Only syslog entries are created by this option. No SNMP traps are issued by this option.

NOTE: RPF can only be configured at the physical port level. It should not be configured on virtual interfaces.

Suppressing RPF for Packets with Specified Address Prefixes

You can suppress RPF packet drops for a specified set of packets using inbound ACLs. To do this:

Create an ACL that identifies the address range that you don't want dropped and specify the flag **suppress-rpf-drop** to the ACL clause. When a packet that fails the RPF check and matches the specified ACL permit clause with the suppress-rpf-drop flag set, it is forwarded as a normal packet and it is accounted as a **unicast RPF suppressed drop** packet as described in Table 13.2.

The following example demonstrates the configuration of **access-list 135** which permits traffic from the source network 4.4.4.0/24 even if the RPF check test fails:

```
IMR 640(config)# access-list 135 permit ip 4.4.4.0.0.0.0.255 any suppress-rpf-drop
IMR 640(config)# access-list 135 permit ip any any
```

Syntax: suppress-rpf-drop

In the following example, access list 135 is applied as an inbound filter on ethernet interface 7/5:

```
IMR 640(config)# interface ethernet 7/5
IMR 640(config-if-e1000-7/5)# rpf-mode strict
IMR 640(config-if-e1000-3/1)# ip access-group 135 in
```

NOTE: If the physical port is a member of a virtual interface, the ACL will have to be applied to the virtual interface instead of the physical port.

Displaying RPF Statistics for the NetIron IMR 640

To display information about RPF configuration and packets that have been dropped because they failed the RPF check, use the **show ip interface** command as shown:

```
NetIron IMR640 Router# show ip interface ethernet 7/1
Interface Ethernet 7/1 (384)
   port enabled
   port state: UP
   ip address: 1.2.3.4/8
   Port belongs to VRF: default
   encapsulation: Ethernet, mtu: 1500
   MAC Address 000c.db24.a6c0
   directed-broadcast-forwarding: disabled
   No inbound ip access-list is set
   No outbound ip access-list is set
   No Helper Addresses are configured
   RPF mode: strict RFP Log: Disabled
   376720 unicast RPF drop 36068 unicast RPF suppressed drop
```

NOTE: The RPF accounting information is always available through the physical interface, even if the physical port belongs to one or more VE's

Table 13.2 describes the RPF statistics displayed when using the show ip interface ethernet command. They are displayed in **bold**. For information about all other parameters, see the *Foundry Enterprise Configuration and Management Guide*.

Table 13.2: RPF Statistics by Port

This Field...	Displays...
RPF Mode:	This display parameter can have one of the following two values: • loose – RPF will permit a packet as long as the source address matches a known route entry in the routing table. It will drop a packet if it doesn't match a route entry. • strict – RPF requires that a packet matches a known route entry as described for loose mode and also that it arrives at the configured interface as described in the router table's next hop information. It will drop a packet that doesn't match both of these criteria.
RPF Log:	This display parameter displays the RPF Log Configuration Status: • Enabled – the RPF log feature has been configured. • Disabled – the RPF log feature has not been configured
<*number*> unicast RPF drop	The number of packets that have been dropped due to failure of the RPF test.
<*number*> unicast RPF suppressed drop	The number of packets would have been dropped due to failure of the RPF test but were not dropped because they matched conditions set in an ACL with the **suppress-rpf-drop** flag set.

Clearing RPF Statistics for a specified Interface

To clear RPF statistics on a specific physical interface use the following command:

Syntax: clear ip interface ethernet <slot/port>

The slot/port variables specify the interface that you want to clear RPF statistics for.

Displaying RPF Logging

If you set the log option of the rpf-mode command, the packets are saved to the system log. To display the log, enter the following:

```
NetIron IMR640 Router#show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 1305 overruns)
    Buffer logging: level ACDMEINW, 50 messages logged
    level code: A=alert C=critical D=debugging M=emergency E=error
                I=informational N=notification W=warning
Dynamic Log Buffer (50 lines):
May 11 12:12:54:I:RPF: Denied 1 packets on port 7/5 tcp 4.4.4.1(0) -> 5.6.7.8(0)
```

NOTE: A maximum of 256 RPF log messages are logged per minute.

Chapter 14
Securing SNMP Access

Simple Network Management Protocol (SNMP) is a set of protocols for managing complex networks. SNMP sends messages, called protocol data units (PDUs), to different parts of a network. SNMP-compliant devices, called agents, store data about themselves in Management Information Bases (MIBs) and return this data to the SNMP requesters.

The chapter “Securing Access to Management Functions” on page 2-1 introduced a few methods used to secure SNMP access. They included the following:

- “Using ACLs to Restrict SNMP Access” on page 2-6
- “Restricting SNMP Access to a Specific IP Address” on page 2-9
- “Restricting SNMP Access to a Specific VLAN” on page 2-11
- “Disabling SNMP Access” on page 2-14

This chapter presents additional methods for securing SNMP access to Foundry devices. It contains the following sections:

- “Establishing SNMP Community Strings” on page 14-1
- “Using the User-Based Security Model” on page 14-5
- “Defining SNMP Views” on page 14-10

Restricting SNMP access using ACL, VLAN, or a specific IP address constitute the first level of defense when the packet arrives at a Foundry device. The next level uses one of the following methods:

- Community string match In SNMP versions 1 and 2
- User-based model in SNMP version 3

SNMP views are incorporated in community strings and the user-based model.

Establishing SNMP Community Strings

SNMP versions 1 and 2 use community strings to restrict SNMP access. The default passwords for Web management access are the SNMP community strings configured on the device.

- The default read-only community string is “public”. To open a read-only Web management session, enter “get” and “public” for the user name and password.
- Beginning with software release 05.0.00, there is no default read-write community string. Thus, by default, you cannot open a read-write management session using the Web management interface. You first must configure a read-write community string using the CLI. Then you can log on using “set” as the user name and

the read-write community string you configure as the password.

You can configure as many additional read-only and read-write community strings as you need. The number of strings you can configure depends on the memory on the device. There is no practical limit.

The Web management interface supports only one read-write session at a time. When a read-write session is open on the Web management interface, subsequent sessions are read-only, even if the session login is “set” with a valid read-write password.

NOTE: If you delete the startup-config file, the device automatically re-adds the default “public” read-only community string the next time you load the software.

NOTE: As an alternative to the SNMP community strings, you can secure Web management access using local user accounts or ACLs. See “Setting Up Local User Accounts” on page 2-19 or “Using an ACL to Restrict Web Management Access” on page 2-6.

Encryption of SNMP Community Strings

The software automatically encrypts SNMP community strings. Users with read-only access or who do not have access to management functions in the CLI cannot display the strings. For users with read-write access, the strings are encrypted in the CLI but are shown in the clear in the Web management interface.

Encryption is enabled by default. You can disable encryption for individual strings or trap receivers if desired. See the next section for information about encryption.

Adding an SNMP Community String

To add a community string, use one of the following methods. When you add a community string, you can specify whether the string is encrypted or clear. By default, the string is encrypted.

USING THE CLI

To add an encrypted community string, enter commands such as the following:

```
BigIron(config)# snmp-server community private rw
BigIron(config)# write memory
```

Syntax: snmp-server community [0 | 1] <string>
ro | rw [view <viewname>] [<standard-acl-name> | <standard-acl-id>]

The <string> parameter specifies the community string name. The string can be up to 32 characters long.

The **ro | rw** parameter specifies whether the string is read-only (**ro**) or read-write (**rw**).

The **0** | **1** parameter affects encryption for display of the string in the running-config and the startup-config file. Encryption is enabled by default. When encryption is enabled, the community string is encrypted in the CLI regardless of the access level you are using. In the Web management interface, the community string is encrypted at the read-only access level but is visible at the read-write access level.

The encryption option can be omitted (the default) or can be one of the following:

- **0** – Disables encryption for the community string you specify with the command. The community string is shown as clear text in the running-config and the startup-config file. Use this option if you do not want the display of the community string to be encrypted.
- **1** – Assumes that the community string you enter is the encrypted form, and decrypts the value before using it.

NOTE: If you want the software to assume that the value you enter is the clear-text form, and to encrypt display of that form, do not enter **0** or **1**. Instead, omit the encryption option and allow the software to use the default behavior.

If you specify encryption option **1**, the software assumes that you are entering the encrypted form of the community string. In this case, the software decrypts the community string you enter before using the value for authentication. If you accidentally enter option **1** followed by the clear-text version of the community string, authentication will fail because the value used by the software will not match the value you intended to use.

The command in the example above adds the read-write SNMP community string "private". When you save the new community string to the startup-config file (using the **write memory** command), the software adds the following command to the file:

```
snmp-server community 1 <encrypted-string> rw
```

To add an non-encrypted community string, you must explicitly specify that you do not want the software to encrypt the string. Here is an example:

```
BigIron(config)# snmp-server community 0 private rw
BigIron(config)# write memory
```

The command in this example adds the string "private" in the clear, which means the string is displayed in the clear. When you save the new community string to the startup-config file, the software adds the following command to the file:

```
snmp-server community 0 private rw
```

The **view** <viewstring> parameter is optional. It allows you to associate a view to the members of this community string. Enter up to 32 alphanumeric characters. If no view is specified, access to the full MIB is granted. The view that you want must exist before you can associate it to a community string. Here is an example of how to use the view parameter in the community string command:

```
BigIron(config)# snmp-s community myread ro view sysview
```

The command in this example associates the view "sysview" to the community string named "myread". The community string has read-only access to "sysview". For information on how create views, see the section "Defining SNMP Views" on page 14-10.

The <standard-acl-name> | <standard-acl-id> parameter is optional. It allows you to specify which ACL group will be used to filter incoming SNMP packets. You can enter either the ACL name or its ID. Here are some examples:

```
BigIron(config) # snmp-s community myread ro view sysview 2
BigIron(config) # snmp-s community myread ro view sysview myacl
```

The command in the first example indicates that ACL group 2 will filter incoming SNMP packets; whereas, the command in the second example uses the ACL group called "myacl" to filter incoming packets. See "Using ACLs to Restrict SNMP Access" on page 2-6 for more information.

USING THE WEB MANAGEMENT INTERFACE

NOTE: To make configuration changes, including changes involving SNMP community strings, you must first configure a read-write community string using the CLI. Alternatively, you must configure another authentication method and log on to the CLI using a valid password for that method.

To use the Web interface to add a community string, do the following:

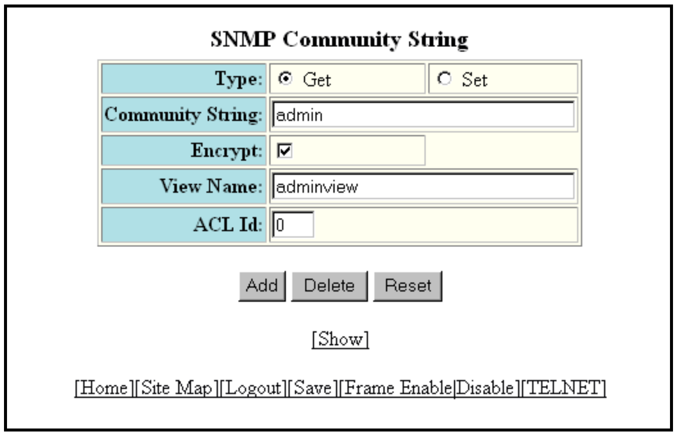
1. Log on to the device using a valid user name and password for read-write access.

   NOTE: If you have configured the device to secure Web management access using local user accounts, you must instead enter the user name and password of one of the user accounts. See "Setting Up Local User Accounts" on page 2-19.

2. Click the Management link on the System configuration panel to display the Management configuration panel.

3. Click the Community String link to display the SNMP Community String panel. This panel shows a list of configured community strings.

   For example,


SNMP Community String
Add
Delete
Reset
[Show]
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

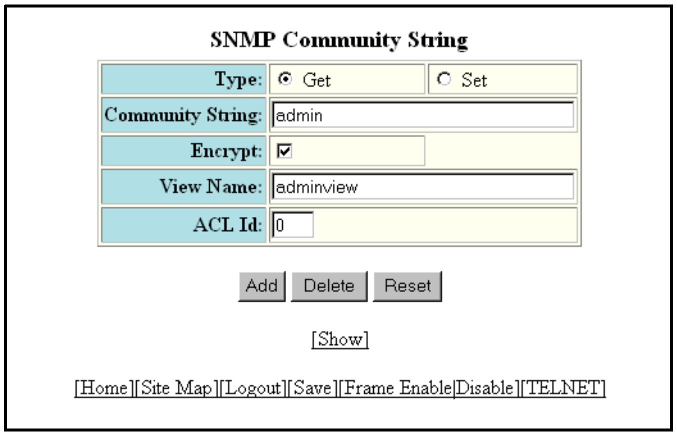
Type:	[✓ Get] Set
Community String:	admin
Encrypt:	✓
View Name:	adminview
ACL Id:	0

4. Click Add Community String to display the SNMP Community String fields.
5. Select the type of community string you are adding by clicking the "Get" or "Set" button. "Get" provides read-only access, while "Set" provides read-write access.
6. Enter the name of the community string.
7. Encryption is enabled by default. Remove the checkmark from the Encrypt box if you want to disable encryption of the string display. If you disable encryption, other users can view the community string.

   To re-enable encryption, place a checkmark in the Encrypt box.
8. Enter a name for the view that will be assigned to the community string.
9. Enter the number of the ACL that will be used to filter SNMP packets for this community string.

NOTE: In this release, ACL by name is not supported in the Web Interface.

Here is an example of a completed form.


SNMP Community String
Add
Delete
Reset
[Show]
[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

Type:	[✓ Get] Set
Community String:	admin
Encrypt:	✓
View Name:	adminview
ACL Id:	0

10. Click Add to apply the change to the device's running-config file.
11. Select the Save link at the bottom of the panel. Select Yes when prompted to save the configuration change to the startup-config file on the device's flash memory.

Displaying the SNMP Community Strings

To display the SNMP community strings, use one of the following methods.

USING THE CLI

To display the configured community strings, enter the following command at any CLI level:

```
BigIron(config)# show snmp server
```

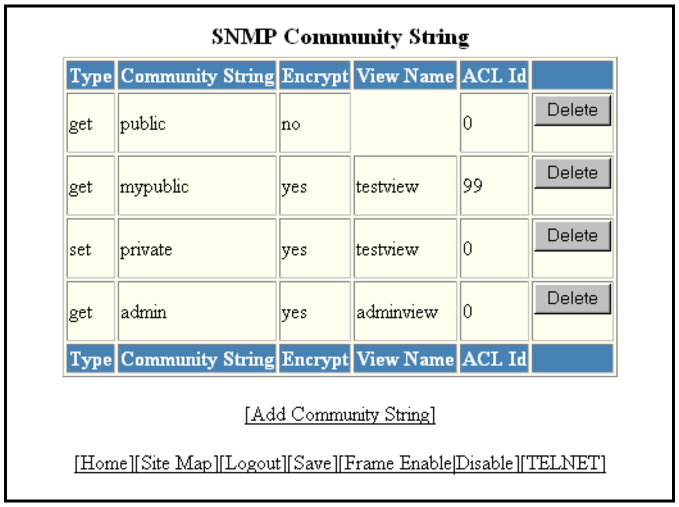
Syntax: show snmp server

See the *Foundry Switch and Router Command Line Interface Reference* for an example of the information displayed by the command.

NOTE: If display of the strings is encrypted, the strings are not displayed. Encryption is enabled by default.

USING THE WEB MANAGEMENT INTERFACE

1. Log on to the device using a valid user name and password for read-write access.
2. Select the Management link from the System configuration panel to display the Management configuration panel.
3. Select the Community String link to display the SNMP Community String panel, as shown in the following example.



SNMP Community String

Type	Community String	Encrypt	View Name	ACL Id	
get	public	no		0	Delete
get	mypublic	yes	testview	99	Delete
set	private	yes	testview	0	Delete
get	admin	yes	adminview	0	Delete
Type	Community String	Encrypt	View Name	ACL Id	

[Add Community String]

[Home][Site Map][Logout][Save][Frame Enable|Disable][TELNET]

Using the User-Based Security Model

SNMP version 3 (RFC 2570 through 2575) introduces a User-Based Security model (RFC 2574) for authentication and privacy services.

SNMP version 1 and version 2 use community strings to authenticate SNMP access to management modules. This method can still be used for authentication. In SNMP version 3, the User-Based Security model of SNMP can be used to secure against the following threats:

- Modification of information
- Masquerading the identity of an authorized entity

- Message stream modification
- Disclosure of information

Furthermore, SNMP version 3 supports View-Based Access Control Mechanism (RFC 2575) to control access at the PDU level. It defines mechanisms for determining whether or not access to a managed object in a local MIB by a remote principal should be allowed. (See the section “Defining SNMP Views” on page 14-10.)

NOTE: SNMP version 3 Notification is not supported at this time. The system will generate traps in SNMP version 1 format, just as in earlier releases.

Configuring Your NMS

To be able to use the SNMP version 3 features:

1. Make sure that your Network Manager System (NMS) supports SNMP version 3.
2. Configure your NMS agent with the necessary users.
3. Configure the SNMP version 3 features in Foundry devices.

Configuring SNMP Version 3 on Foundry Devices

To configure SNMP version 3 on Foundry devices, do the following:

1. Enter an engine ID for the management module using the **snmp-server engineid** command if you will not use the default engine ID. See “Defining the Engine ID” on page 14-6.
2. Create views that will be assigned to SNMP user groups using the **snmp-server view** command. See the “Defining SNMP Views” on page 14-10 for details.
3. Create ACL groups that will be assigned to SNMP user groups using the **access-list** command. Refer to the *Foundry Switch and Router Command Line Interface Reference* for details.
4. Create user groups using the **snmp-server group** command. See “Defining an SNMP Group” on page 14-7.
5. Create user accounts and associate these accounts to user groups using the **snmp-server user** command. See “Defining an SNMP User Account” on page 14-8.

NOTE: In this release, configuration of SNMP version 3 features is done using the CLI. No Web Interface or SNMP interface is available.

If SNMP version 3 is not configured, then community strings by default are used to authenticate access.

Defining the Engine ID

A default engine ID is generated during system start up. To determine what the default engine ID of the device is, enter the **show snmp engineid** command and find the following line.

```
Local SNMP Engine ID: 800007c70300e05290ab60
```

See the section “Displaying the Engine ID” on page 14-9 for details.

The default engine ID guarantees the uniqueness of the engine ID for SNMP version 3. If you want to change the default engine ID, enter a command such as the following:

```
BigIron(config)# snmp-server engineid local 800007c70300e05290ab60
```

Syntax: [no] snmp-server engineid local <hex-string>

The **local** parameter indicates that engine ID to be entered is the ID of this device, representing an SNMP management entity.

NOTE: Since the current implementation of SNMP version 3 does not support Notification, remote engine IDs cannot be configured at this time.

The <hex-string> variable consists of 11 octets, entered as hexadecimal values. There are two hexadecimal characters in each octet. There should be an even number of hexadecimal characters in an engine ID.

The default engine ID has a maximum of 11 octets:

- Octets 1 through 4 represent the agent's SNMP management private enterprise number as assigned by the Internet Assigned Numbers Authority (IANA). The most significant bit of Octet 1 is "1". For example, “000007c7” is the ID for Foundry Networks in hexadecimal. With Octet 1 always equal to "1", the first four octets in the default engine ID is always “800007c7” (which is 1991 in decimal).
- Octet 5 is always 03 in hexadecimal and indicates that the next set of values represent a MAC address.
- Octets 6 through 11 form the MAC address of the lowest port in the management module.

NOTE: Engine ID must be a unique number among the various SNMP engines in the management domain. Using the default engine ID ensures the uniqueness of the numbers.

Defining an SNMP Group

SNMP groups map SNMP users to SNMP views. For each SNMP group, you can configure a read view, a write view, or both. Users who are mapped to a group will use its views for access control.

To configure an SNMP user group, enter a command such as the following:

```
BigIron(config)# snmp-server group admin v3 auth read all write all
```

Syntax: [no] snmp-server group <groupname>
v1 | v2 | v3
auth | noauth | priv
[access <standard-acl-id>] [read <viewstring> | write <viewstring>]

NOTE: This command is not used for SNMP version 1 and SNMP version 2. In these versions, groups and group views are created internally using community strings. (See “Establishing SNMP Community Strings” on page 14-1.) When a community string is created, two groups are created, based on the community string name. One group is for SNMP version 1 packets, while the other is for SNMP version 2 packets.

The **group** <groupname> parameter defines the name of the SNMP group to be created.

The **v1**, **v2**, or **v3** parameter indicates which version of SNMP is used. In most cases, you will be using v3, since groups are automatically created in SNMP versions 1 and 2 from community strings.

The **auth** | **noauth** parameter determines whether or not authentication will be required to access the supported views. If auth is selected, then only authenticated packets are allowed to access the view specified for the user group. Selecting **noauth** means that no authentication is required to access the specified view. Selecting **priv** means that an authentication password will be required from the users.

The **access** <standard-acl-id> parameter is optional. It allows incoming SNMP packets to be filtered based on the standard ACL attached to the group.

The **read** <viewstring> | **write** <viewstring> parameter is optional. It indicates that users who belong to this group have either read or write access to the MIB.

The <viewstring> variable is the name of the view to which the SNMP group members have access. If no view is specified, then the group has no access to the MIB.

The value of <viewstring> is defined using the **snmp-server view** command. The SNMP agent comes with the "all" view, the default view that provides access to the entire MIB; however, it must be specified when creating the group. The "all" view also allows SNMP version 3 to be backwards compatibility with SNMP version 1 and version 2.

NOTE: Previous to the Enterprise software release 08.0.00, the default SNMP group is “v1default”.

NOTE: If you will be using a view other than the "all" view, that view must be configured before creating the user group. See the section “Defining SNMP Views” on page 14-10, especially for details on the include | exclude parameters.

Defining an SNMP User Account

The **snmp-server user** command does the following:

- Creates an SNMP user.
- Defines the group to which the user will be associated.
- Defines the type of authentication to be used for SNMP access by this user.

Here is an example of how to create the account:

```
BigIron(config)# snmp-s user bob admin v3 access 2 auth md5 bobmd5 priv des bobdes
```

The CLI for creating SNMP version 3 users has been updated as follows.

Syntax: [no] snmp-server user <name> <groupname> v3
[[access <standard-acl-id>] [encrypted] [auth md5 <md5-password> | sha <sha-password>]
[priv [encrypted] des <des-password>]]

The <name> parameter defines the SNMP user name or security name used to access the management module.

The <groupname> parameter identifies the SNMP group to which this user is associated or mapped. All users must be mapped to an SNMP group. Groups are defined using the **snmp-server group** command.

NOTE: The SNMP group to which the user account will be mapped should be configured before creating the user accounts; otherwise, the group will be created without any views. Also, ACL groups must be configured before configuring user accounts.

The **v3** parameter is required.

The **access** <standard-acl-id> parameter is optional. It indicates that incoming SNMP packets are filtered based on the ACL attached to the user account.

NOTE: The ACL specified in a user account overrides the ACL assigned to the group to which the user is mapped. If no ACL is entered for the user account, then the ACL configured for the group will be used to filter packets.

The **encrypted** parameter means that the MD5 or SHA password will be a digest value. MD5 has 16 octets in the digest. SHA has 20. The digest string has to be entered as a hexadecimal string. In this case, the agent need not generate any explicit digest. If the **encrypted** parameter is not used, the user is expected to enter the authentication password string for MD5 or SHA. The agent will convert the password string to a digest, as described in RFC 2574.

The **auth md5 | sha** parameter is optional. It defines the type of encryption that the user must have to be authenticated. Choose between MD5 or SHA encryption. MD5 and SHA are two authentication protocols used in SNMP version 3.

The <md5-password> and <sha-password> define the password the user must use to be authenticated. These password must have a minimum of 8 characters. If the encrypted parameter is used, then the digest has 16 octets for MD5 or 20 octets for SHA.

NOTE: Once a password string is entered, the generated configuration displays the digest (for security reasons), not the actual password.

The **priv [encrypted] des** <des-password> parameter is optional. It defines the type of encryption that will be used to encrypt the privacy password. If the "encryption" keyword is used, enter a 16-octet DES key in hexadecimal format for the des-password. If the "encryption" keyword is not used enter a password string. The agent will generate a suitable 16-octet DES key from the password string.

Currently, DES is the only encryption type supported for priv password.

Displaying the Engine ID

To display the engine ID of a management module, enter a command such as the following:

```
BigIron(config)# show snmp engineid
Local SNMP Engine ID: 800007c70300e05290ab60
Engine Boots: 3
Engine time: 5
```

Syntax: show snmp engineid

The engine ID identifies the source or destination of the packet.

The engine boots represents the number of times that the SNMP engine reinitialized itself with the same engine ID. If the engineID is modified, the boot count is reset to 0.

The engine time represents the current time with the SNMP agent.

Displaying SNMP Groups

To display the definition of an SNMP group, enter a command such as the following:

```
BigIron(config)# show snmp group
groupname = exceptifgrp
security model = v3
security level = authNoPriv
ACL id = 2
readview = exceptif
writeview = <none>
```

Syntax: show snmp group

The value for security level can be one of the following:

Security Level	Authentication
<none>	If the security model shows v1 or v2, then security level is blank. User names are not used to authenticate users; community strings are used instead.
noauthNoPriv	Displays if the security model shows v3 and user authentication is by user name only.
authNoPriv	Displays if the security model shows v3 and user authentication is by user name and the MD5 or SHA algorithm.

Displaying User Information

To display the definition of an SNMP user account, enter a command such as the following:

```
BigIron(config)# show snmp user

username = bob
acl id = 2
group = admin
security model = v3
group acl id = 0
authtype = md5
authkey = 3aca18d90b8d172760e2dd2e8f59b7fe
privtype = des,  privkey = 1088359afb3701730173a6332d406eec
engine ID= 800007c70300e052ab0000
```

Syntax: show snmp user

Interpreting Varbinds in Report Packets

If an SNMP version 3 request packet is to be rejected by an SNMP agent, the agent sends a report packet that contains one or more varbinds. The varbinds contain additional information, showing the cause of failures. An SNMP manager application decodes the description from the varbind. The following table presents a list of varbinds supported by the SNMP agent.

Varbind Object Identifier	Description
1. 3. 6. 1. 6. 3. 11. 2. 1. 3. 0	Unknown packet data unit.
1. 3. 6. 1. 6. 3. 12. 1. 5. 0	The value of the varbind shows the engine ID that needs to be used in the snmp-server engineid command
1. 3. 6. 1. 6. 3. 15. 1. 1. 1. 0	Unsupported security level.
1. 3. 6. 1. 6. 3. 15. 1. 1. 2. 0	Not in time packet.
1. 3. 6. 1. 6. 3. 15. 1. 1. 3. 0	Unknown user name. This varbind may also be generated: • If the configured ACL for this user filters out this packet. • If the group associated with the user is unknown.
1. 3. 6. 1. 6. 3. 15. 1. 1. 4. 0	Unknown engine ID. The value of this varbind would be the correct authoritative engineID that should be used.
1. 3. 6. 1. 6. 3. 15. 1. 1. 5. 0	Wrong digest.
1. 3. 6. 1. 6. 3. 15. 1. 1. 6. 0	Decryption error.

Defining SNMP Views

SNMP views are named groups of MIB objects that can be associated with user accounts to allow limited access for viewing and modification of SNMP statistics and system configuration. SNMP views can also be used with other commands that take SNMP views as an argument. SNMP views reference MIB objects using object names, numbers, wildcards, or a combination of the three. The numbers represent the hierarchical location of the object in the MIB tree. You can reference individual objects in the MIB tree or a subset of objects from the MIB tree.

To configure the number of SNMP views available on the Foundry device:

```
BigIron(config)# system-max view 15
```

Syntax: system-max view <number-of-views>

This command specifies the maximum number of SNMPv2 and v3 views that can be configured on a device. The number of views can be from 10 – 65536. The default is 10 views.

To add an SNMP view, enter one of the following commands:

```
BigIron(config)# snmp-server view Maynes system included
BigIron(config)# snmp-server view Maynes system.2 excluded
BigIron(config)# snmp-server view Maynes 2.3.*.6 included
BigIron(config)# write mem
```

NOTE: The **snmp-server view** command supports the MIB objects as defined in RFC 1445.

Syntax: [no] snmp-server view <name> <mib_tree> included | excluded

The <name> parameter can be any alphanumeric name you choose to identify the view. The names cannot contain spaces.

The <mib_tree> parameter is the name of the MIB object or family. MIB objects and MIB sub-trees can be identified by a name or by the numbers called Object Identifiers (OIDs) that represent the position of the object or sub-tree in the MIB hierarchy. You can use a wildcard (*) in the numbers to specify a sub-tree family.

The **included** | **excluded** parameter specifies whether the MIB objects identified by the <mib_family> parameter are included in the view or excluded from the view.

NOTE: All MIB objects are automatically excluded from any view unless they are explicitly included; therefore, when creating views using the **snmp-server view** command, indicate which portion of the MIB you want users to access.

For example, you may want to assign the view called “admin” a community string or user group. The “admin” view will allow access to the Foundry MIBs objects that begin with the 1.3.6.1.4.1.1991 object identifier. Enter the following command:

```
BigIron(config)# snmp-server view admin 1.3.6.1.4.1.1991 included
```

You can exclude portions of the MIB within an inclusion scope. For example, if you want to exclude the snAgentSys objects, which begin with 1.3.6.1.4.1.1991.1.1.2 object identifier from the admin view, enter a second command such as the following:

```
BigIron(config)# snmp-server view admin 1.3.6.1.4.1.1991.1.1.2 excluded
```

Note that the exclusion is within the scope of the inclusion.

To delete a view, use the no parameter before the command.

Configuration Examples

The following sections present examples of how to configure SNMP v3.

Simple SNMP v3 Configuration

```
BigIron(config)#snmp-s group admingrp v3 priv read all write all notify all
BigIron(config)#snmp-s user adminuser admingrp v3 auth md5 <auth password> priv
<privacy password>
BigIron(config)#snmp-s host <dest-ip> version v3 privacy adminuser
```

More Detailed SNMP v3 Configuration

```
BigIron(config)#snmp-server view internet internet included
BigIron(config)#snmp-server view system system included
BigIron(config)#snmp-server community ..... ro
BigIron(config)#snmp-server community ..... rw
BigIron(config)#snmp-server contact isc-operations
BigIron(config)#snmp-server location sdh-pillbox
BigIron(config)#snmp-server host 128.91.255.32 .....
BigIron(config)#snmp-server group ops v3 priv read internet write system
BigIron(config)#snmp-server group admin v3 priv read internet write internet
BigIron(config)#snmp-server group restricted v3 priv read internet
BigIron(config)#snmp-server user ops ops v3 encrypted auth md5
ab8e9cd6d46e7a270b8c9549d92a069 priv encrypted des 0e1b153303b6188089411447dbc32de
BigIron(config)#snmp-server user admin admin v3 encrypted auth md5
0d8a2123f91bfbd8695fef16a6f4207b priv encrypted des
18e0cf359fce4fcd60df19c2b6515448
BigIron(config)#snmp-server user restricted restricted v3 encrypted auth md5
261fd8f56a3ad51c8bcec1e4609f54dc priv encrypted des
d32e66152f89de9b2e0cb17a65595f43
```