Foundry Security Guide

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

November 2002

Contents

Chapter 1
Getting Started

Introduction

This guide describes how to secure access to management functions on a Foundry device. In addition, this guide explains how to protect Foundry devices from Denial of Service (DoS) attacks.

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, IS-IS, BGP4, MBGP, IGMP, PIM, DVMRP, IPX, AppleTalk, FSRP, VRRP, and VRRPE.

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 hardware and software installation information, and configuration information for system-level features.
- *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.
- *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.

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.

What's New In This Edition?

This edition describes the following software release:

- 07.6.01

This release applies to the following products:

- NetIron Internet Backbone router
- BigIron with M2 (Management II) or higher modules
- BigIron with Velocity Management Module version I (VM1)
- FastIron II, FastIron II Plus, and FastIron III with M2 or higher modules
- FastIron 4802

For a list of the enhancements, see the "Getting Started" chapter in the *Foundry Switch and Router Installation and Basic Configuration Guide*.

NOTE: If you want documentation specifically for a 07.1.x release, see the January, 2001 edition of the manuals and the release notes for the release you are using. For the 07.2.06 release or a 07.3.x release, see the June, 2001 edition and the release notes.

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 2-1 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 2-3 explains how to restrict access to management functions from remote sources, including Telnet, the Web management interface, and SNMP
- "Setting Passwords" on page 2-10 explains how to set passwords for Telnet access and management privilege levels
- "Setting Up Local User Accounts" on page 2-14 explains how to define user accounts to regulate who can access management functions
- "Configuring TACACS/TACACS+ Security" on page 2-16 explains how to configure SNMP read-only and read-write community strings on a Foundry device
- "Configuring TACACS/TACACS+ Security" on page 2-16 explains how to configure TACACS/TACACS+ authentication, authorization, and accounting
- "Configuring RADIUS Security" on page 2-33 explains how to configure RADIUS authentication, authorization, and accounting
- "Configuring Authentication-Method Lists" on page 2-49 explains how to set the order that authentication methods are consulted when more than one is used with an access method

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	2-11

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
Access to the Privileged EXEC and CONFIG levels of the CLI	Not secured	Establish a password for Telnet access to the CLI	2-10
		Establish passwords for management privilege levels	2-11
		Set up local user accounts	2-14
		Configure TACACS/TACACS+ security	2-16
		Configure RADIUS security	2-33
Telnet access	Not secured	Regulate Telnet access using ACLs	2-4
		Allow Telnet access only from specific IP addresses	2-6
		Allow Telnet access only to clients connected to a specific VLAN	2-7
		Disable Telnet access	2-8
		Establish a password for Telnet access	2-10
		Establish passwords for privilege levels of the CLI	2-11
		Set up local user accounts	2-14
		Configure TACACS/TACACS+ security	2-16
		Configure RADIUS security	2-33
Secure Shell (SSH) access	Not configured	Configure SSH	3-1
		Regulate SSH access using ACLs	2-4
		Establish passwords for privilege levels of the CLI	2-11
		Set up local user accounts	2-14
		Configure TACACS/TACACS+ security	2-16
		Configure RADIUS security	2-33

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
Web management access	SNMP read or read-write community strings	Regulate Web management access using ACLs	2-5
		Allow Web management access only from specific IP addresses	2-6
		Allow Web management access only to clients connected to a specific VLAN	2-7
		Disable Web management access	2-8
		Set up local user accounts	2-14
		Establish SNMP read or read-write community strings for SNMP versions 1 and 2	8-1
		Establishing user groups for SNMP version 3	8-5
		Configure TACACS/TACACS+ security	2-16
		Configure RADIUS security	2-33
SNMP (IronView) 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	2-5
		Allow SNMP access only from specific IP addresses	2-6
		Disable SNMP access	2-10
		Allow SNMP access only to clients connected to a specific VLAN	2-7
		Establish passwords to management levels of the CLI	2-11
		Set up local user accounts	2-14
		Establish SNMP read or read-write community strings	2-16
TFTP access	Not secured	Allow TFTP access only to clients connected to a specific VLAN	2-7

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.

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 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.

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 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) 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 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 or third-party SNMP management applications.

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:

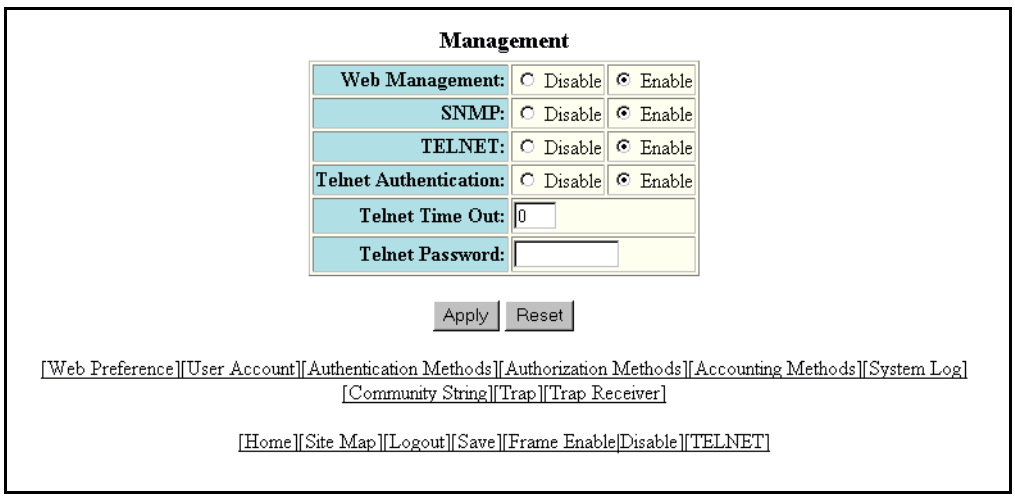
```
BigIron(config)# web-management
```

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.

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



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 SNMP Access

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

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.

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 2-10.
- 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 2-11.

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 2-14.

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 2-14.

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 2-49.

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.

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 2-13.

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 2-11.

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 2-49.

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>

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.

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

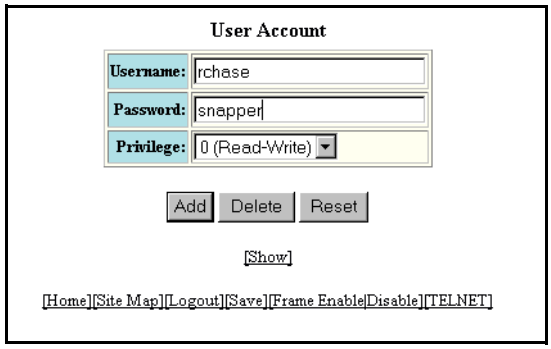
USING THE WEB MANAGEMENT INTERFACE

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 2-13.

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



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 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 (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: In releases prior to 07.1.00, a user logging into the device via Telnet or SSH would first enter the User EXEC level. The user could then enter the **enable** command to get to the Privileged EXEC level.

Starting with release 07.1.00, a user that is successfully authenticated by a RADIUS or TACACS+ server is automatically placed at the Privileged EXEC level after login.

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 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 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 authorization for **logout** command (TACACS+): aaa authorization commands <privilege-level> default <method-list>
	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 Action	Applicable AAA Operations
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 2-21.
2. Set optional parameters. See "Setting Optional TACACS/TACACS+ Parameters" on page 2-22.
3. Configure authentication-method lists. See "Configuring Authentication-Method Lists for TACACS/TACACS+" on page 2-23.

TACACS+ Configuration Procedure

For TACACS+ configurations, use the following procedure:

1. Identify TACACS+ servers. See "Identifying the TACACS/TACACS+ Servers" on page 2-21.

2. Set optional parameters. See “Setting Optional TACACS/TACACS+ Parameters” on page 2-22.
3. Configure authentication-method lists. See “Configuring Authentication-Method Lists for TACACS/TACACS+” on page 2-23.
4. Optionally configure TACACS+ authorization. See “Configuring TACACS+ Authorization” on page 2-25.
5. Optionally configure TACACS+ accounting. See “Configuring TACACS+ Accounting” on page 2-27.

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 2-23.) 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 2-10.
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 2-11.
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 2-14.
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 2-49.

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 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.

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.

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

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 chassis 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
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

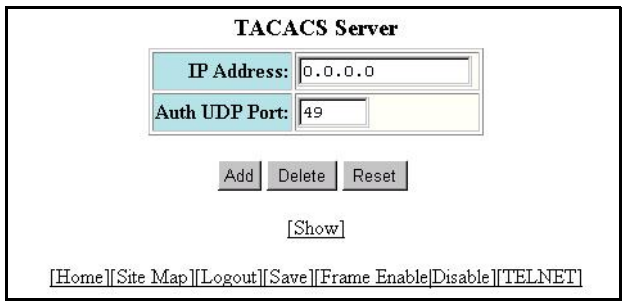
To configure TACACS/TACACS+ using the Web management interface:

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.

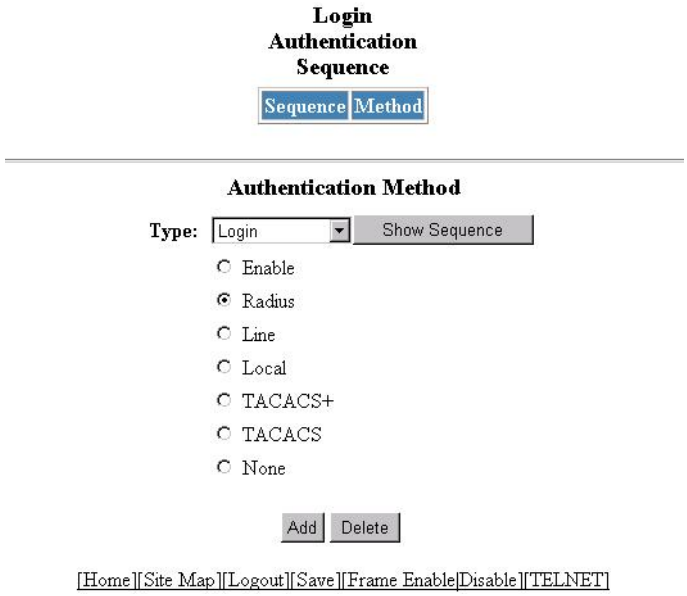
---

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.



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.

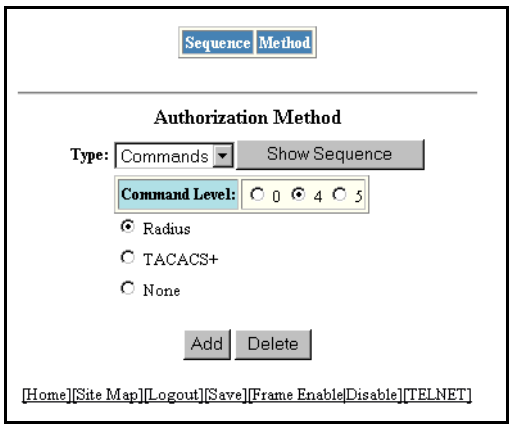


17. Select the type of access for which you are defining the authentication method list from the Type field's pulldown menu. Each type of access must have a separate authentication-method list. For example, to define the authentication-method list for logging into the CLI, select Login.
18. Select the primary authentication method by clicking on the radio button next to the method. For example, to use a 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.



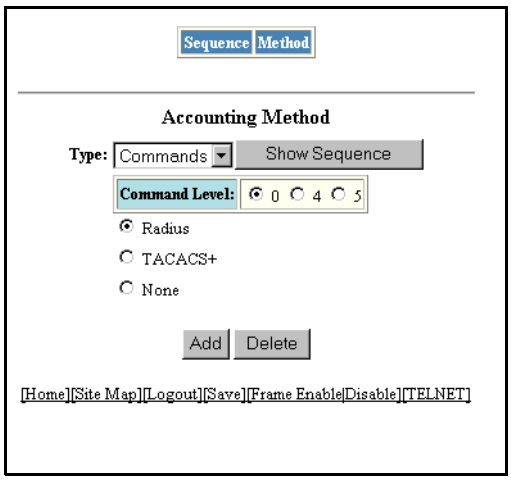
22. To configure TACACS+ exec authorization, select Exec from the Type field's pulldown menu.
23. To configure TACACS+ command authorization, select Commands from the Type field's pulldown menu and select a privilege level by clicking on one of the following radio buttons:
    - **0** – Authorization 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.
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.



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) 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: In releases prior to 07.1.00, a user logging into the device via Telnet or SSH would first enter the User EXEC level. The user could then enter the **enable** command to get to the Privileged EXEC level.

Starting with release 07.1.00, a user that is successfully authenticated by a RADIUS or TACACS+ server is automatically placed at the Privileged EXEC level after login.

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 2-37.
2. Identify the RADIUS server to the Foundry device. See "Identifying the RADIUS Server to the Foundry Device" on page 2-38.
3. Set RADIUS parameters. See "Setting RADIUS Parameters" on page 2-39.
4. Configure authentication-method lists. See "Configuring Authentication-Method Lists for RADIUS" on page 2-40.
5. Optionally configure RADIUS authorization. See "Configuring RADIUS Authorization" on page 2-42.
6. Optionally configure RADIUS accounting. "Configuring RADIUS Accounting" on page 2-43.

Configuring Foundry-Specific Attributes on the RADIUS Server

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 2-10.
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 2-11.
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 2-14.
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 2-49.

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. When RADIUS 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 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 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.

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

NOTE: A user's privilege level is set during RADIUS authentication, not with an **aaa authorization** command. The command **aaa authorization exec default radius** is ignored by the system.

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

WARNING: 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 Chassis 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.

Table 2.7: Output of the show aaa command for RADIUS

Field	Description
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

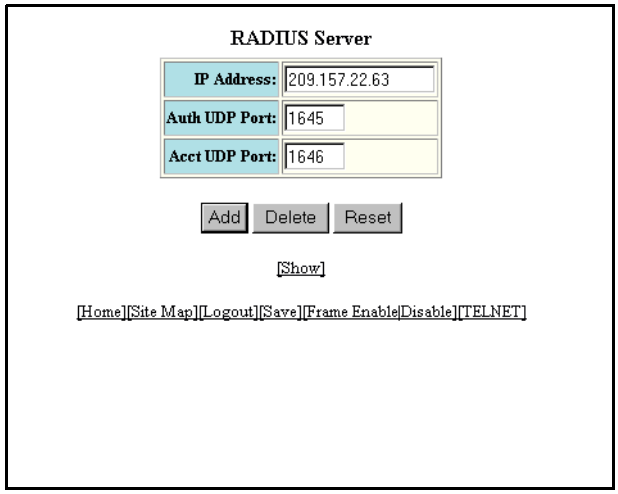
USING THE WEB MANAGEMENT INTERFACE

To configure RADIUS using the Web management interface:

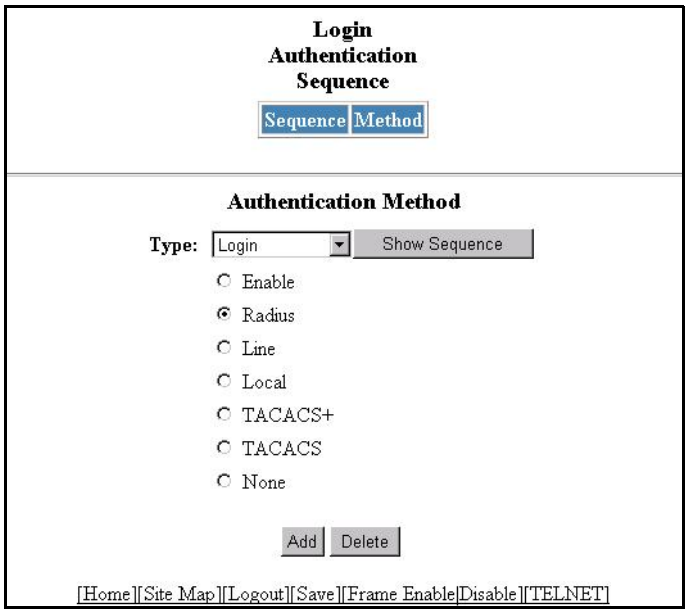
1. Log on to the device using a valid user name and password for read-write access. The System configuration panel is displayed.
2. If you configuring RADIUS authentication for Telnet access to the CLI, go to step 3. Otherwise, go to step 7.
3. Select the Management link to display the Management configuration 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.



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.

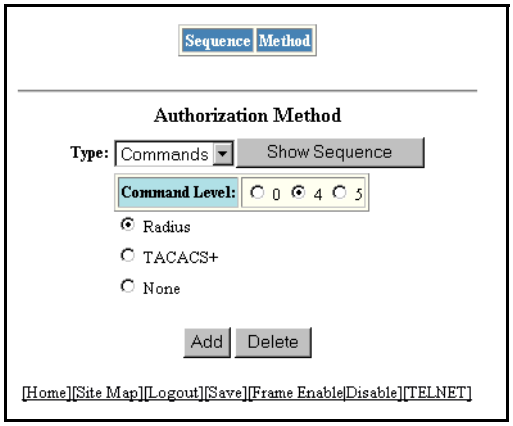


17. Select the type of access for which you are defining the authentication method list from the Type field's pulldown menu. Each type of access must have a separate authentication-method list. For example, to define the authentication-method list for logging into the CLI, select Login.

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.

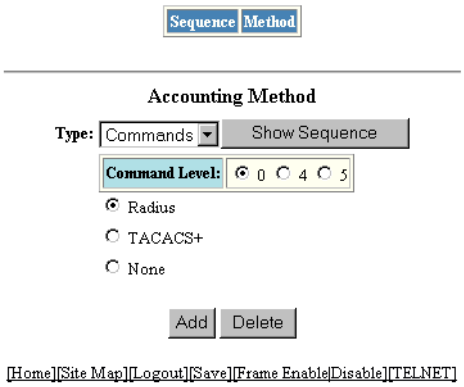
NOTE: Since RADIUS authorization relies on the command list supplied by the RADIUS server during authentication, you cannot perform RADIUS authorization without RADIUS authentication.

NOTE: A user's privilege level is set during RADIUS authentication, not by configuring RADIUS Exec authorization. Selecting RADIUS Exec authorization on the Authorization Method panel is ignored by the system.

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 2-6.

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.

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 2-16.
- 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, 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 is not authenticated. To use local user accounts to authenticate access through IronView, 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, 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, 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 IronView.

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

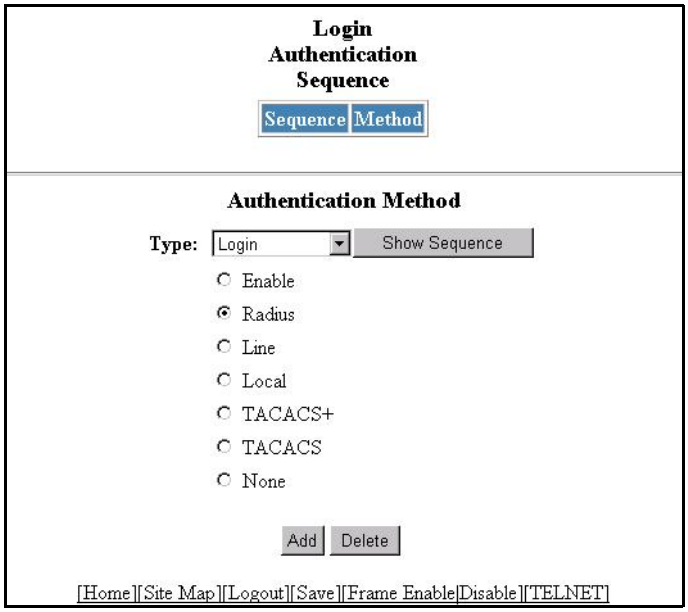
Method Parameter	Description
line	Authenticate using the password you configured for Telnet access. The Telnet password is configured using the **enable telnet password…** command. See "Setting a Telnet Password" on page 2-10.
enable	Authenticate using the password you configured for the Super User privilege level. This password is configured using the **enable super-user-password…** command. See "Setting Passwords for Management Privilege Levels" on page 2-11.
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 2-14.
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 2-33.
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.



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.

Chapter 3
Configuring Secure Shell

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 3-10 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)
- FastIron Workgroup Layer 2 Switch (8MB models only, switch code only)

NOTE: Foundry's implementation of SSH supports SSH version 1 only. All references to SSH in this document are to SSH version 1.

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.

Configuring Secure Shell on a Foundry device consists of the following steps:

1. Setting the Foundry device's host name and domain name
2. Generating a host RSA public and private key pair for the device

3. Configuring RSA challenge-response authentication
4. Setting 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 RSA 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 3-3 for an example of what to place in the known hosts file.

To generate a public and private RSA host key pair for the Foundry device:

```
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, 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 rsa** 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 rsa** command, the new pair is stored in internal memory the first time you enter the **write memory** command.
- The **crypto key zeroize rsa** 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.
- 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.

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 11877188186267703046485128873725804685603164063588767923018424702263617580489663338462057493006839765023169898543185727932374596324079021803229084221453472515782437007702806627934784079949643404159653290224014833380339095421473679746385600601629453293075635028042310396543882204328326628042425693615834281633 1
```

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
949573392322599631573796819248476346145327421786527672319957469414416047146826800
064453679033330420291249056907718288654183965655676902543288147725297813592782l
675406294783926622751287748618154485239970236181733123284766607218888739467582O1
 user@csp_client
1024 35 152676199889856769693556155614587291553826312328095300428421494164360924
762074755452346792684432337622953129794188335259756957757051018052125410080748777
265861198574227028970041121688521450740879698406424084517427145585923616937059080
748378755994055034796030242871313127938950079274380749727874236959776352519431 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 3-10 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>

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

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


 1024 65537 162566050678380006149460550286514061230306797782065166110686648548574
 949573392322599631573796819248476346145327421786527672319957469414416047146826800
 064453679033330420291249056907718288654183965655676902543288147725297813592782l
 675406294783926622751287748618154485239970236181733123284766607218888739467582O1
  user@csp_client

 1024 35 152676199889856769693556155614587291553826312328095300428421494164360924
 762074755452346792684432337622953129794188335259756957757051018052125410080748777
 265861198574227028970041121688521450740879698406424084517427145585923616937059080
 748378755994055034796030242871313127938950079274380749727874236959776352519430 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 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 3-7). 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 2-14 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.

Viewing 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
411931951252058944526374624095222755050208450873029852099603462391729956763293 57
247775301886662678981956482531815516246813945206816726108281883104139622423012 96
268839371767697761849840931009840170753693870710066379666508772246779794868026 51
458324218055083313313948534902409 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 3-3 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 802.1X Port Security

Overview

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 sub-network.

This chapter is divided into the following sections:

- "How 802.1X Port Security Works" on page 4-1 explains basic concepts about 802.1X port security.
- "Configuring 802.1X Port Security" on page 4-6 describes how to set up 802.1X port security on Foundry devices using the Command Line Interface (CLI).
- "Displaying 802.1X Information" on page 4-11 describes the commands used to display information about an 802.1X port security configuration.
- "Sample 802.1X Configurations" on page 4-15 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.

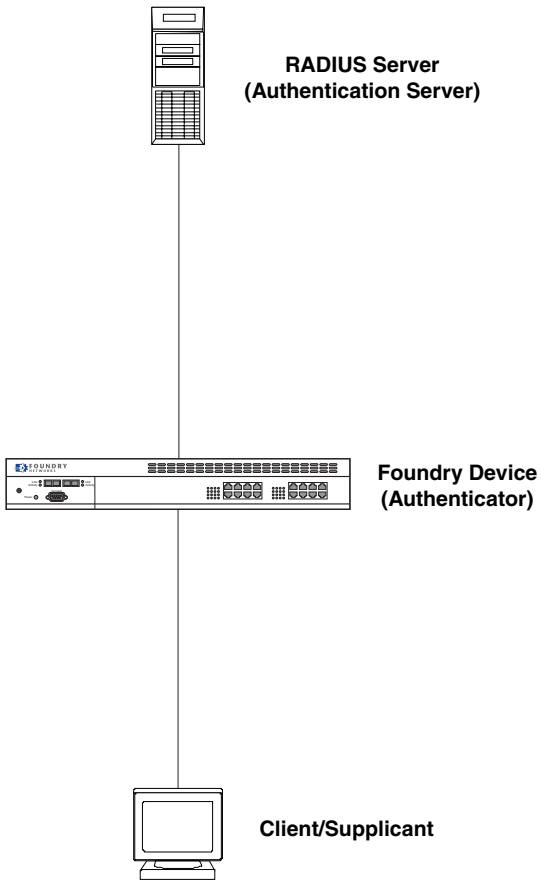
Device Roles in an 802.1X Configuration

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 4.1 illustrates these roles.

Figure 4.1 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.

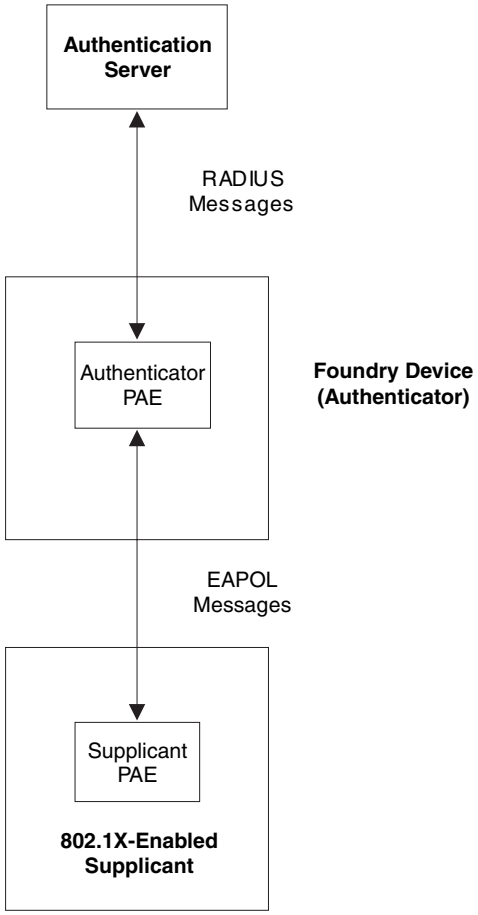
Authentication Server – The device that validates the Client and specifies whether or not the Client may access services on the device. Foundry supports Authentication Servers running RADIUS.

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 ***EAPOL*** (EAP over LAN). 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 4.2 shows the relationship between the Authenticator PAE and the Supplicant PAE.

Figure 4.2 Authenticator PAE and Supplicant PAE



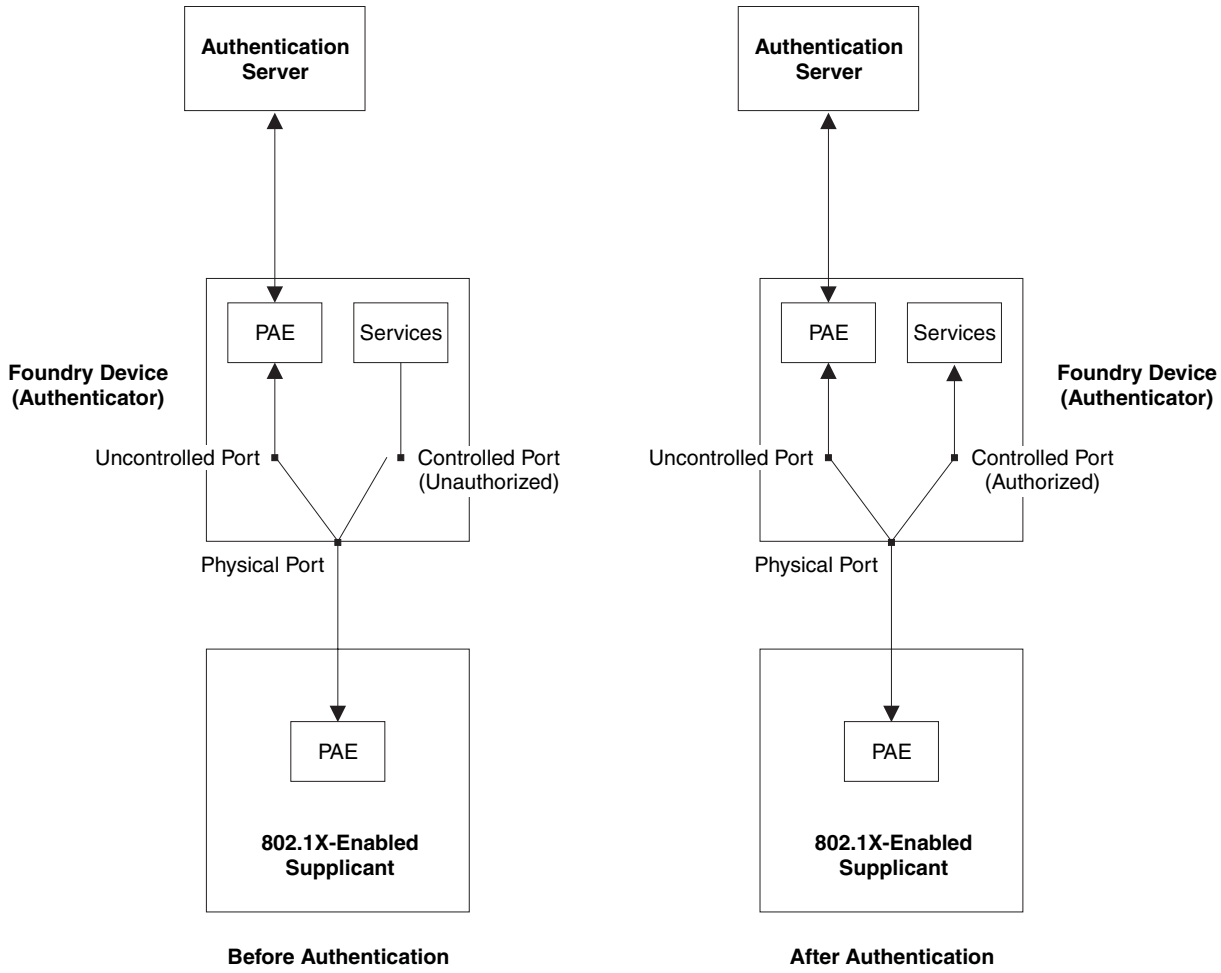
Authenticator PAE – The Authenticator PAE communicates with the Supplicant PAE, receiving identifying information from the Supplicant. Acting as a RADIUS client, the Authenticator PAE passes the Supplicant's information to the Authentication Server, which decides whether the Supplicant can gain access 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 4.3 illustrates this concept.

Figure 4.3 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.

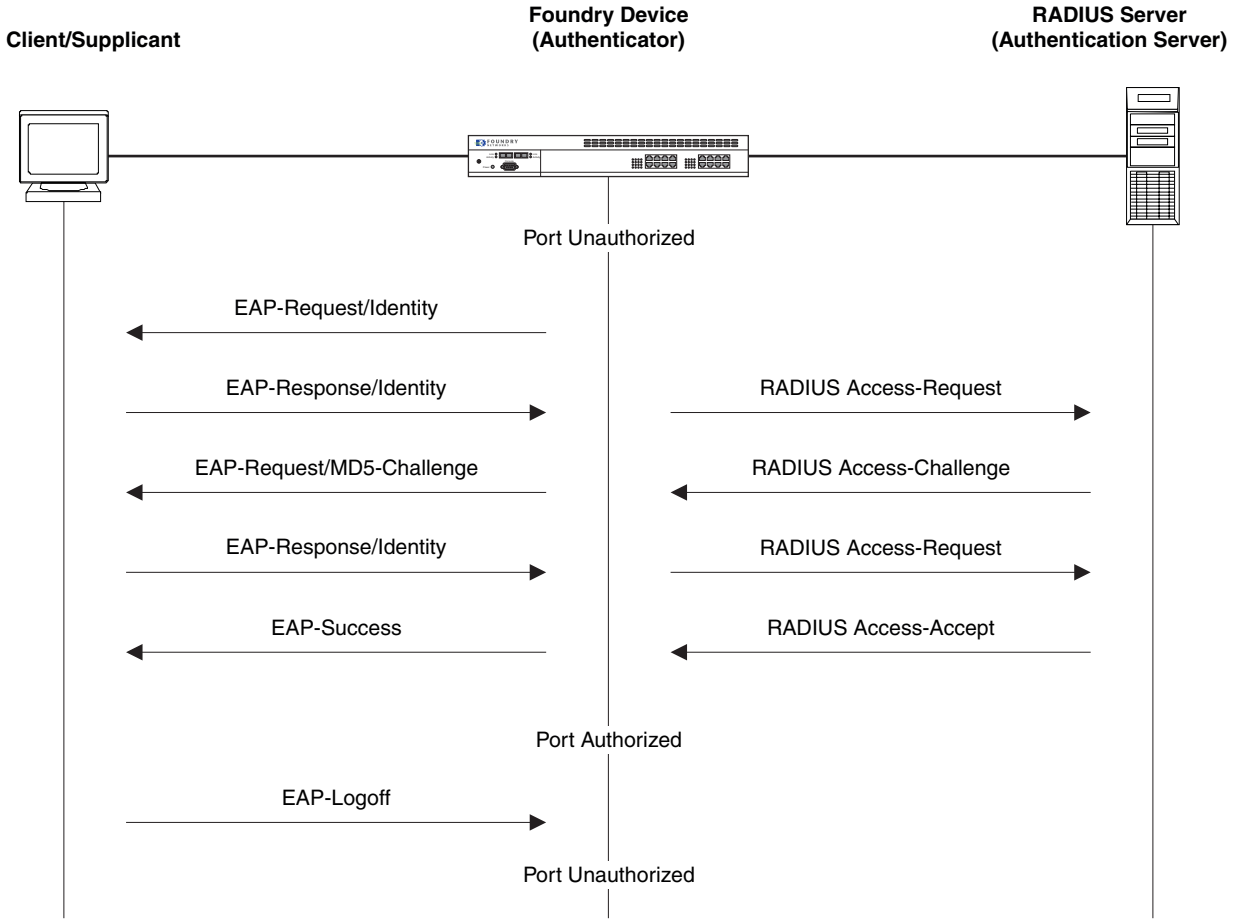
During authentication, EAPOL messages are exchanged between the Supplicant PAE and the Authenticator PAE, and RADIUS messages are exchanged between the Authenticator PAE and the Authentication Server. See "Message Exchange During Authentication" on page 4-4 for an example of this process. If the Client 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 4-7 for more information.

Message Exchange During Authentication

Figure 4.4 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 4.4 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.

If the 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.

NOTE: Foundry devices support Identity and MD5-challenge request types in EAP Request/Response messages.

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 4-6
   - “Setting RADIUS Parameters” on page 4-6
2. Configuring the Foundry device’s role as the Authenticator:
   - “Enabling 802.1X Port Security” on page 4-7
   - “Initializing 802.1X on a Port” on page 4-10 (optional)
3. Configuring the Foundry device’s interaction with Clients:
   - “Configuring Periodic Re-Authentication” on page 4-8 (optional)
   - “Re-Authenticating a Port Manually” on page 4-8 (optional)
   - “Setting the Quiet Period” on page 4-9 (optional)
   - “Setting the Interval for Retransmission of EAP-Request/Identity Frames” on page 4-9 (optional)
   - “Specifying the Number of EAP-Request/Identity Frame Retransmissions” on page 4-9 (optional)
   - “Specifying the Security Hold Time” on page 4-9 (optional)
   - “Specifying a Timeout for Retransmission of EAP-Request Frames to the Client” on page 4-10 (optional)
   - “Allowing Access to Multiple Hosts” on page 4-10 (optional)
   - “Defining MAC Filters for EAP Frames” on page 4-11 (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
```


Syntax: radius-server host <ip-addr> | <server-name> [auth-port <number> acct-port <number> default key <string> dot1x]

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

The **default key** <string> **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.

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 4.3 on page 4-4 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:

- Link aggregation
- Metro Ring Protocol (MRP)
- MAC port security
- Tagged port
- Mirror port
- Route only
- Trunk port

In addition, you cannot configure 802.1X port security on a virtual interface (ve).

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>

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>

Setting the Interval for Retransmission of EAP-Request/Identity Frames

When the Foundry device sends a Client an EAP-request/identity frame, it expects to receive an EAP-response/identity frame from the Client. If the Client does not send back an EAP-response/identity frame, the device waits a specified amount of time and then retransmits the EAP-request/identity frame. You can specify the amount of time the Foundry device waits before retransmitting the EAP-request/identity frame to the Client. This amount of time is specified with the **tx-period** parameter. The **tx-period** parameter can be from 0 – 4294967295 seconds. The default is 30 seconds.

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
```

Syntax: [no] timeout tx-period <seconds>

If the Client does not send back an EAP-response/identity frame within 60 seconds, the device will transmit another EAP-request/identity frame.

Specifying the Security Hold Time

The **multiple-hosts** command (see “Allowing Access to Multiple Hosts” on page 4-10) allows more than one 802.1X Client to connect on an interface. However, when the **multiple-hosts** command is not used in a 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.

Specifying the Number of EAP-Request/Identity Frame Retransmissions

If the Foundry device does not receive a EAP-response/identity frame from a Client, the device waits 30 seconds (or the amount of time specified with the **timeout tx-period** command), then retransmits the EAP-request/identity frame. By default, 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, the device restarts the authentication process with the Client.

You can optionally 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>

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 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>

Specifying a Timeout for Retransmission of EAP-Request Frames to the Client

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. 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. The time constraint for retransmission of EAP-Request frames to the Client can be between 0 – 4294967295 seconds.

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>

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

In a multiple-host configuration, the Foundry device is connected to a hub that has more than one device connected to it. To enable 802.1X port security in this kind of configuration, the Foundry device 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 4.6 on page 4-17 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.

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.

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.

Displaying 802.1X Information

You can display information about the 802.1X configuration on the device and on individual ports, as well as statistics about the EAPOL frames passing through the device.

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
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 4.1: 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 4-8. When periodic re-authentication is enabled, the device automatically re-authenticates Clients every 3,600 seconds by default.

Table 4.1: Output from the show dot1x command (Continued)

This Field...	Displays...
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 4-9 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 Interval for Retransmission of EAP-Request/Identity Frames” on page 4-9 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 “Specifying a Timeout for Retransmission of EAP-Request Frames to the Client” on page 4-10 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 4-10 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 “Specifying the Number of EAP-Request/Identity Frame Retransmissions” on page 4-9 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 4-8 for information on how to change this setting.
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 4-9 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 3/1
Port 3/1 Configuration:
Authenticator PAE state:    CONNECTING
Backend Authentication state:     IDLE
AdminControlledDirections:    BOTH
```



```
OperControlledDirections:    BOTH
AuthControlledPortControl:     Auto
AuthControlledPortStatus:    unauthorized
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
Protocol Version:    1
```

Syntax: show dot1x config <portnum>

The following additional information is displayed in the **show dot1x config** command for an interface:

Table 4.2: 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.
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.

Table 4.2: Output from the show dot1x config command for an interface (Continued)

This Field...	Displays...
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 4-10 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 4.3: 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.

Table 4.3: Output from the show dot1x statistics command (Continued)

This Field...	Displays...
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>

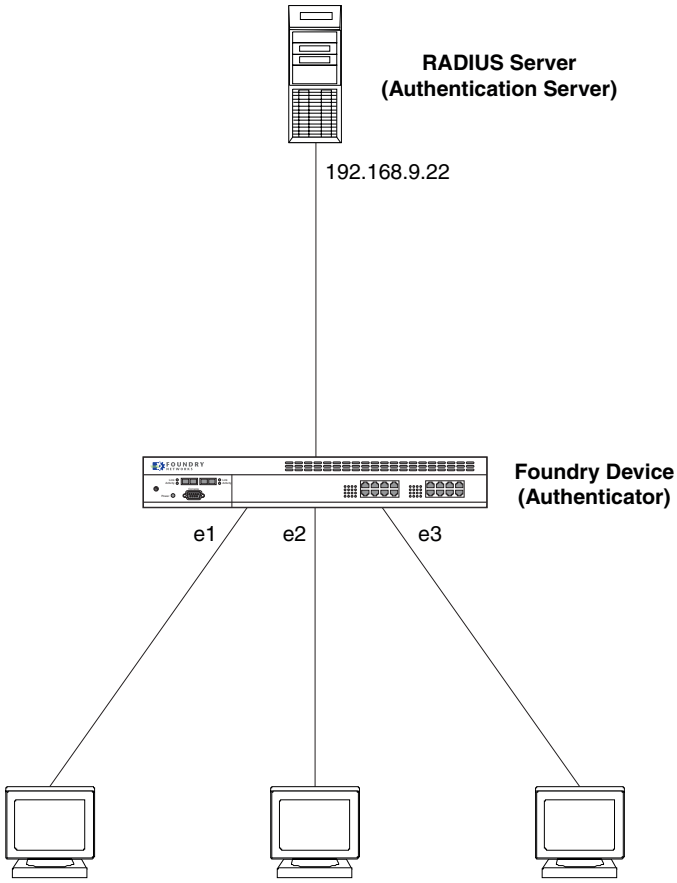
Sample 802.1X Configurations

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 4.5 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 4.5 Sample point-to-point 802.1X configuration



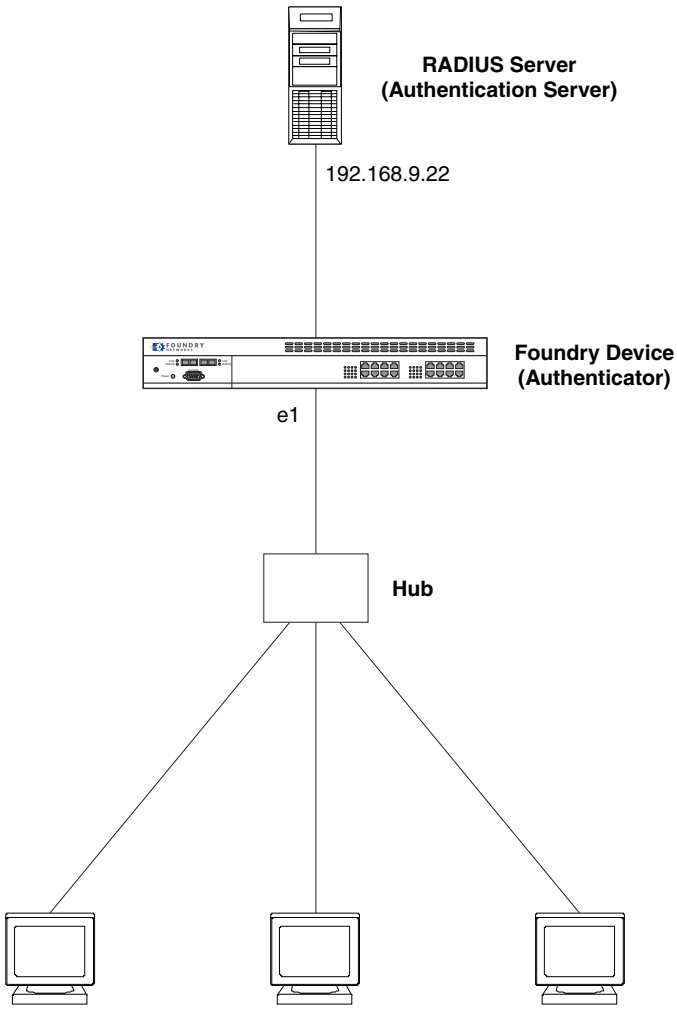
The following commands configure the Foundry device in Figure 4.5:

```
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

Figure 4.6 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 4.5, 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 4.6 Sample 802.1X configuration using a hub



The following commands configure the Foundry device in Figure 4.6:

```
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
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
```


Chapter 5
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.

Configuring the Port Security Feature

To configure the port security feature, you perform the following tasks:

- Enable the 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

Enabling the Port Security Feature

By default, the 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.

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 5.1: 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 5.2: 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 5-1 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 5.3: 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 5.4: 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 6
Protecting Against Denial of Service Attacks

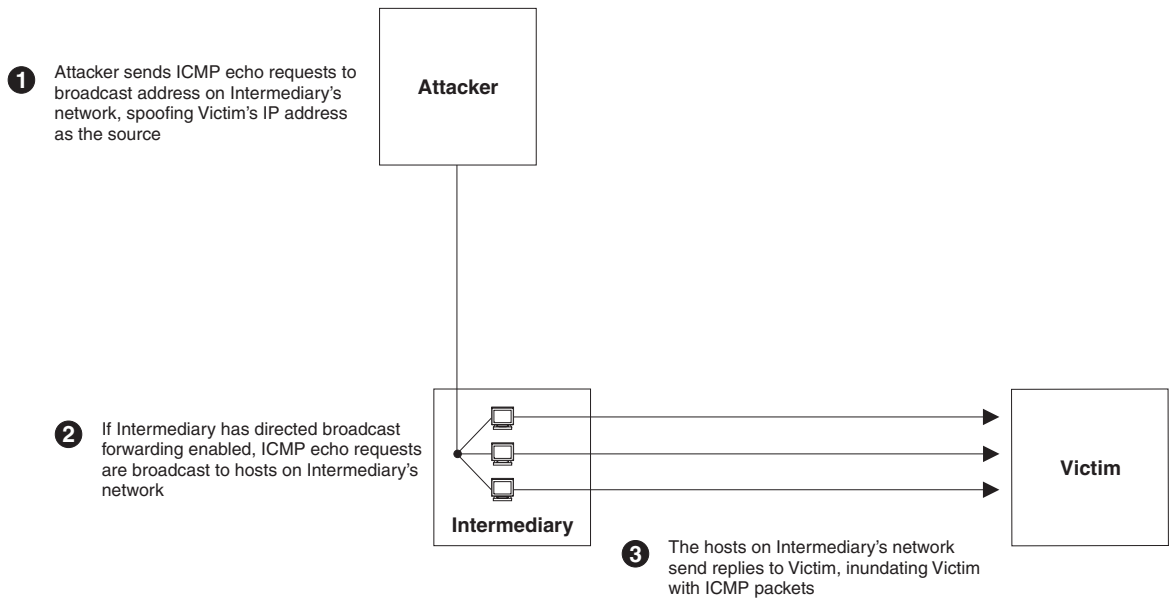
In a Denial of Service (DoS) attack, a router is flooded with useless packets, hindering normal operation. Foundry devices include measures for defending against two types of DoS attacks: Smurf attacks and TCP SYN attacks.

NOTE: For information about configuring the ServerIron to defend against TCP SYN attacks, see "ServerIron DoS Attack Protection" on page 7-1.

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 6.1 illustrates how a Smurf attack works.

Figure 6.1 How a Smurf attack floods a victim with ICMP replies



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 sub-net. When the ICMP echo request packet arrives at the target sub-net, 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. The command is available at the interface level only on Chassis devices. 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).

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

Chapter 7
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 6-2. 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 7-4.
- 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 7-4.

NOTE: The SYN-Defense and SYN-Guard features are available on the ServerIron 400 and ServerIron 400 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
```

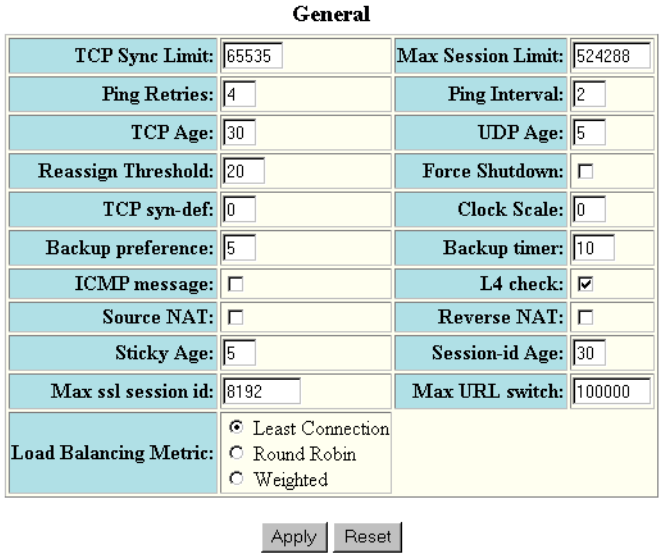
Syntax: show server traffic

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:



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.

NOTE: SYN-Defense is supported on the ServerIron 400 and ServerIron 800 only

When a connecting client sends a TCP SYN to a server, the ServerIron 400 or ServerIron 800 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 400 or ServerIron 800 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 400 or ServerIron 800 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 400 or ServerIron 800 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 400 or ServerIron 800, 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 400 and ServerIron 800 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 400 or ServerIron 800 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

SYN-Guard™

The SYN-Guard feature on the ServerIron 400 and ServerIron 800 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 the ServerIron 400 or ServerIron 800. 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 400 or ServerIron 800 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.

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]

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 8
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 2-5
- "Restricting SNMP Access to a Specific IP Address" on page 2-6
- "Restricting SNMP Access to a Specific VLAN" on page 2-7
- "Disabling SNMP Access" on page 2-10

This chapter presents additional methods for securing SNMP access to Foundry devices. It contains the following sections:

- "Establishing SNMP Community Strings" on page 8-1
- "Using the User-Based Security Model" on page 8-5
- "Defining SNMP Views" on page 8-10

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 2-14 or "Using an ACL to Restrict Web Management Access" on page 2-5.

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 8-10.

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 2-5 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:

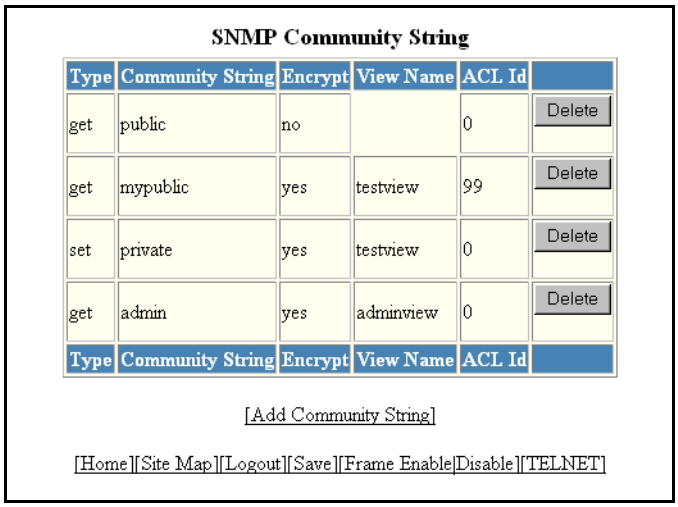
1. Log on to the device using a valid user name and password for read-write access.

   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 2-14.

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,

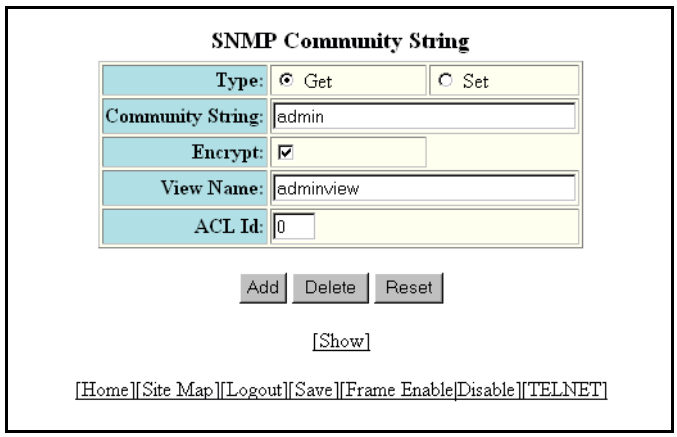


4. Click Add Community String to display the SNMP Community String fields.
5. Select the type of community string you are adding by clicking the "Get" or "Set" button. "Get" provides read-only access, while "Set" provides read-write access.
6. Enter the name of the community string.
7. Encryption is 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.



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
```

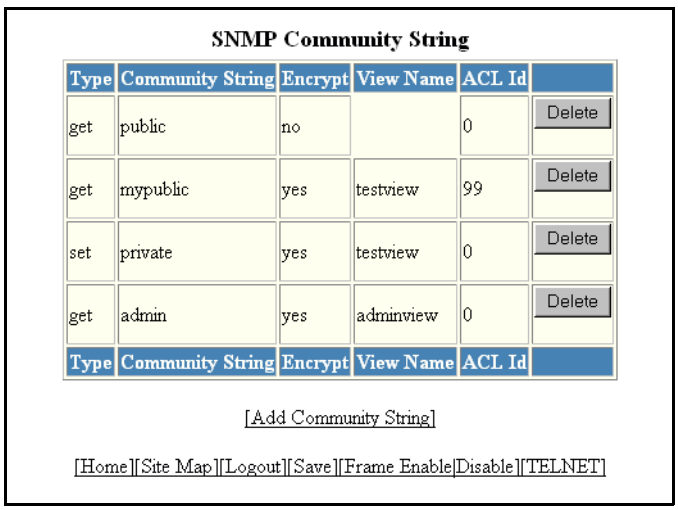
Syntax: show snmp server

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.



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 8-10.)

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 8-6.
2. Create views that will be assigned to SNMP user groups using the **snmp-server view** command. See the "Defining SNMP Views" on page 8-10 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 8-7.
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 8-8.

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 8-9 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 v1default write v1default
```

Syntax: [no] snmp-server group <groupname>
v1 | v2 | v3
auth | noauth
[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 8-1.) 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.

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 "v1default" view, the default view that provides access to the entire MIB; however, it must be specified when creating the group. The "v1default" view also allows SNMP version 3 to be backwards compatibility with SNMP version 1 and version 2.

NOTE: If you will be using a view other than the "v1default" view, that view must be configured before creating the user group. See the section “Defining SNMP Views” on page 8-10, 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.

Index

