ARX® CLI Network-Management Guide

Publication Date

This manual was published on September 13, 2012.

Legal Notices

Any modifications to this device, unless expressly approved by the manufacturer, can void the user's authority to operate this equipment under part 15 of the FCC rules.

Canadian Regulatory Compliance

This Class A digital apparatus complies with Canadian ICES-003.

Standards Compliance

This product conforms to the IEC, European Union, ANSI/UL and Canadian CSA standards applicable to Information Technology products at the time of manufacture.

Acknowledgments

This product includes software from several third-party vendors. Each vendor is listed below with applicable copyright.

Revision History

September 2006 - Rev A

October 2006 - Rev B, updates for Software Release 2.4.2

March 2007 - Rev C, updates for Software Release 2.5.0

May 2007 - Rev D, updates for Software Release 2.5.1

November 2007 - Rev E, new Security-Table Appendix for Software Release 2.7.0

December 2007 - Rev F, updates for Software Release 3.0.0

March 2008 - Rev G, updates for Software Release 3.1.0; convert to F5 format

June 2008 - Rev H, updates for Software Release 4.0.0

October 2008 - Rev J, re-brand the OS

June 2009 - Rev K, updates for Software Release 5.00.000

November 2009 - Rev L, updates for Software Release 5.01.000

January 2010 - Rev M, changes for Release 5.01.000 update

March 2010 - Rev N, updates for Software Release 5.01.005

July 2010 - Rev P, updates for Software Release 5.02.000

January 2011 - Rev Q. updates for Software Release 5.03.000

June 2011 - Rev R, updates for Software Release 6.00.000

September 2011 - Rev S, updates for Software Release 6.01.000

July 2012 - Rev T, updates for Software Release 6.02.000
October 2012 - Rev U, updates for Software Release 6.03.000

Table of Contents

1
Introduction

2
Managing Administrative Accounts

3
Configuring Layer 2

4
Configuring the Network Layer

7
Adding a Redundant Switch

8

Configuring Management Access

A

CLI Security Levels

1

Introduction

This manual contains instructions and best practices for setting up and maintaining the Adaptive Resource Switch (ARX®) in your layer-2 and IP network. These instructions focus on the Command-Line Interface (CLI), which provides the advanced features of networking. For basic networking instructions, you can use the GUI's network-setup wizard.

The ARX

The Adaptive Resource Switch (ARX®) is a highly available and scalable solution that brings resource awareness to a file storage infrastructure, and adapts these resources to meet the demands of users and applications in real time. The ARX provides a file-virtualization layer that aggregates the total capacity and performance of your file storage. A *namespace* provides location-independent, transparent mapping of user requests onto the appropriate storage resource. You can configure policies that the switch enforces for the placement, replication and migration of files. Through policy configuration, the ARX adapts to the real-time demands of users and applications. The ARX thereby serves as a *resource proxy* for the files and services behind it.

Back-end Storage and Servers

The Adaptive Resource Switch aggregates heterogeneous file systems and storage into a unified pool of file storage resources. Through this unification, you can manage these resources to adapt to user demands and client applications. File storage assets can be differentiated based on user-defined attributes, enabling a class-of-storage model. You can reclaim stranded capacity through policy implementation for more effective storage utilization, and you can add capacity without disruption. Back-end resources are monitored for availability and performance, as well as user-access patterns that drive policy decisions.

Front-end Services

The Adaptive Resource Switch acts as an in-band file proxy for the Network File System (NFS) and Microsoft's Common Internet File System (CIFS) protocols.
Front-end services provide the file virtualization layer that masks the physical file storage from the user and application. The switch becomes the file access point, as opposed to the actual physical resource, providing file access through a namespace. Users and applications maintain a single consistent file path that is transparently mapped to the proper physical resource where the information resides.

Policy

The Adaptive Resource Switch provides policy-based resource switching. Through *policy* configuration, you can optimize the placement of files onto the appropriate storage resources and automatically adapt these resources based on user and application demand. The ARX performs file replication and migration based on performance, usage or other life-cycle characteristics, enabling you to implement a flexible file services strategy.

Examples of policies include: migrating files to reclaim stranded capacity; migrating files across different tiers of storage based on access patterns and/or value; and replicating frequently accessed files for performance. The result is more efficient utilization and greater flexibility in file storage management.

Resilient Overlay Network (RON)

You can connect multiple ARXes with a Resilient Overlay Network (RON), which can reside on top of any IP network. This provides a network for distributing and accessing file storage. ARXes can replicate storage to other switches in the same RON, updating the replicas periodically as the writable master files change. This is called a *shadow copy*, where a source volume on one switch periodically copies its files to one or more *shadow volumes* on other switches. Clients can access the shadow volumes at multiple geographic locations, independent to where the source volume resides.

Audience for this Manual

This manual is intended for

- network technicians responsible for layer 1 and 2 networks,
- network engineers responsible for the Internet Protocol (IP) layer (layer 3),
- storage engineers who design and manage storage systems (SANs, NASes, and DASes), and
- crypto officers who manage all of the Critical Security Parameters (CSPs) of a network.

The text presumes that all readers are comfortable with a command-line interface (CLI), especially one based on the Cisco IOS.

Using this Manual

This manual contains instructions to set up and maintain networking and administration on a new ARX. Before you begin, you must follow the instructions in your *Hardware Installation Guide* to install the switch, set up its management IP, and prepare it for CLI and/or GUI provisioning. Then you can follow the order of the chapters in this manual to

1. set up administrative users and groups,
2. configure layer-2,
3. configure layer-3, the network layer,
4. join with other switches to form a RON, and possibly.
5. join with another switch to form a redundant pair.

The final chapter has instructions to configure security for management access (for example, to allow other users to access the CLI and/or the GUI).

You can later return to any chapter to update the configuration at a particular layer.

Document Conventions

This manual uses the following conventions:

`this font` represents screen input and output;

- **`bold text`** represents input, and
- *`italic text`* appears for variable input or output.

this font is used for command-syntax definitions, which use the same rules for bold and italic.

Command-syntax definitions also use the following symbols:

- [*optional-argument*] - square brackets ([ ]) surround optional arguments;
- *choice1* | *choice2* - the vertical bar (|) separates argument choices;
- {*choice1* | *choice2* | *choice3*} - curly braces ({ }) surround a required choice;
- [*choice1* | *choice2*]* - an asterisk (*) means that you can choose none of them, or as many as desired (for example, "choice1 choice2" chooses both);
- {*choice1* | *choice2*}+ - a plus sign (+) means that you must choose one or more.

CLI Overview

The Command-Line Interface (CLI) has its commands grouped into modes.

Modes are structured as a tree with a single root, *exec* mode. This section summarizes the mode structure and explains some CLI conventions.

Exec Mode

When you log into the CLI, you begin in exec mode. If the hostname is "bstnA," the prompt appears as shown below:

```
bstnA>
```

You can access all global commands (such as show commands) from exec mode, and you can use the enable command to enter priv-exec mode.

```
bstnA> enable
```

Global Commands

You can access global commands from any mode, not just exec. Global commands include all show commands and terminal-control commands.

Priv-exec Mode

Priv-exec mode has the following prompt:

```
bstnA#
```

Priv-exec mode contains chassis-management commands, clock commands, and other commands that require privileges but do not change the network or storage configuration.

Priv-exec has two sub modes, cfg and gbl.

Cfg Mode

To enter cfg mode, use the config command:

```
bstnA# config
bstnA(cfg)#
```

Config mode contains all modes and commands for changing the configuration of the local switch, such as network configuration.

Gbl Mode

To enter gbl mode, use the global command:

```
bstnA# global
bstnA(gbl)#
```

Gbl mode controls all parameters that are shared in a redundant pair, such as namespaces and global servers.

Exiting a Mode

From any mode, use the exit command to return to its parent mode. From priv-exec mode, this command exits the CLI; to go from priv-exec mode back to exec mode, use the no enable command.

From any submode of cfg or gbl mode, you can return immediately to priv-exec mode by using the end command or pressing <Ctrl-z>.

Prompts

Prompts contain information about your position in the mode hierarchy as well as the name of the object you are configuring. For example, suppose you use the following command in gbl mode:

```
bstnA(gbl)# namespace wwmed
bstnA(gbl-ns[wwmed])#
```

This command places you into a new mode, as indicated by the new CLI prompt. The prompt shows the name of the mode, "gbl-ns," and the name of the configuration object, a namespace called "wwmed." Abbreviations are used for mode names (for example, "ns" instead of "namespace") to conserve space on the command line.

When you descend to lower modes in the config tree, the prompt offers more information. To extend the previous example, suppose you enter the following command to configure the "/local" volume in the wwmed namespace:

```
bstnA(gbl-ns[wwmed])# volume /local
bstnA(gbl-ns-vol[wwmed~/local])#
```

The tilde character (~) separates a parent object from its child: "wwmed~/local" shows that you are in the "/local" volume under the "wwmed" namespace.

The no Convention

Most config commands have the option to use the "no" keyword to negate the command. For commands that create an object, the no form removes the object. For commands that change a default setting, the no form reverts back to the default. As an example,

```
bstnA(gbl-ns[wwmed])# no volume /local
```

removes the "/local" volume from the "wwmed" namespace.

The enable/no enable Convention

Many objects and configurations require you to enable them using the enable command before they can take effect. Likewise, many objects and configurations require you to first disable them using the no enable command before you can complete a related command or function. The no

enable command does not remove an object; it only disables it until you re-enable it. The enable/no enable commands exist in many modes and submodes in the CLI.

For example, the following command sequence enables the namespace named “wwmed:”

```
bstnA(gbl)# namespace wwmed
bstnA(gbl-ns[wwmed])# enable
bstnA(gbl-ns[wwmed])# ...
```

Getting Started

For the initial login, refer to the instructions for booting and configuring the switch in the appropriate *Hardware Installation Guide*.

For subsequent logins, use the following steps to log into the F5 CLI:

1. If you are on-site, you can connect a serial line to the serial console port. This port is labeled 'Console' or '10101' (depending on your ARX platform). By default, the port is set for 9600 baud, 8, N, 1.

   You can also telnet to the switch's management interface. For example:

   ```
   telnet 10.10.10.10
   ```

   In either case, a login prompt appears:

```
Username:
```

2. Enter your username and password. For example:

```
Username: admin
Password: acopia
```

   The CLI prompt appears:

```
SWITCH>
```

   The name, "SWITCH," is the default hostname. The hostname is reset as part of the initial-boot process, so it is likely that yours will differ.

Entering Cfg or Gbl Mode

The CLI contains two major configuration modes: cfg and gbl. The cfg mode contains submodes for configuring locally-scoped parameters, only applicable to the local ARX. These parameters include layer-2, layer-3, and chassis configuration. Gbl mode applies to configuration that is shared among both switches in a redundant pair, such as namespaces and global servers.

After you log into the CLI, use the config command to enter cfg mode:

```
SWITCH> enable
SWITCH# configure
SWITCH(cfg)#
```

To enter gbl mode, use the global command instead:

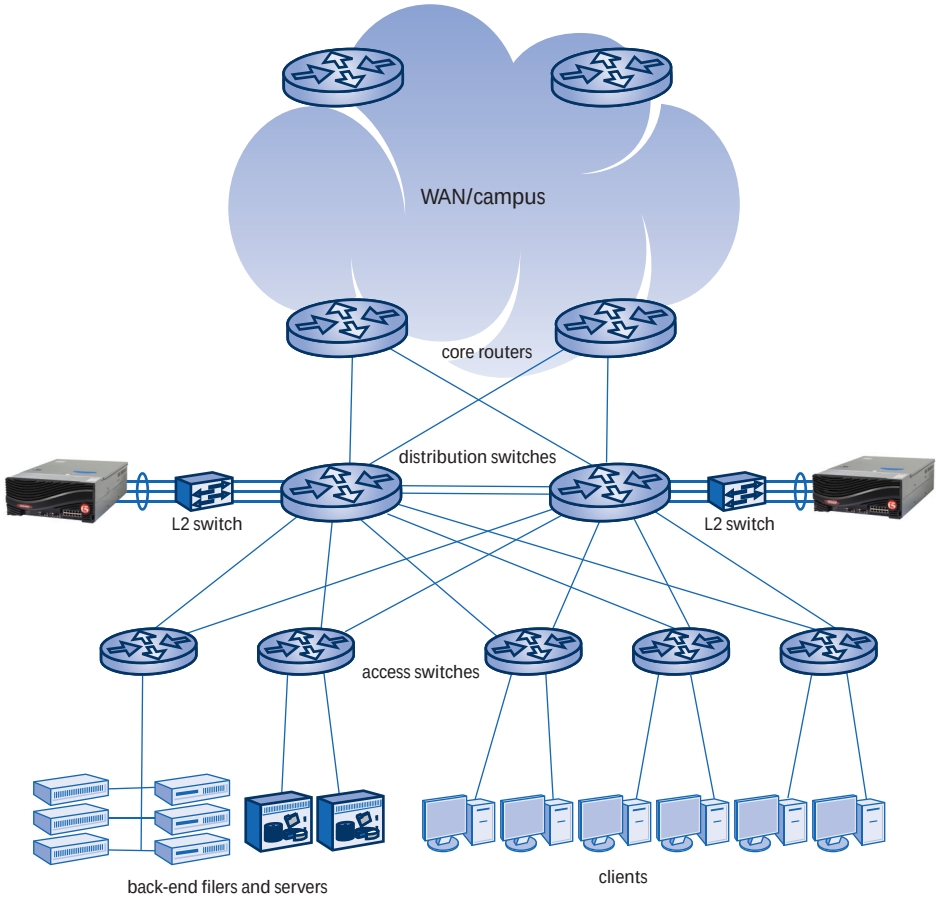
```
SWITCH> enable
SWITCH# global
SWITCH(gbl)#
```

The command sequences in this manual all begin either in cfg mode or gbl mode.

Sample Network

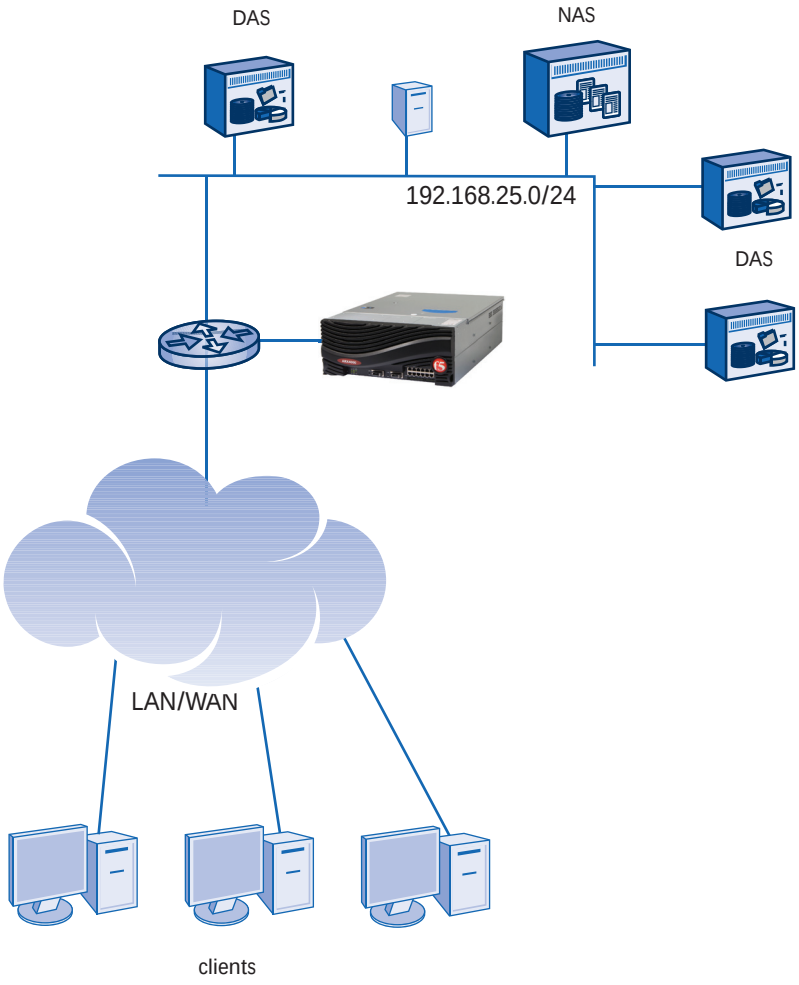
The examples in this manual draw from a single, fictitious network. This section shows the topology of the network and the placement of the ARX.

The environment is assumed to be a three-tiered network in a large data center. The first tier is core routers that provide connectivity to a campus or WAN. The second tier has redundant distribution switches that distribute all data-center traffic between the access switches; the access switches constitute the third tier of the network. All LAN clients, back-end servers, and back-end filers connect to the access switches, which are layer-2 bridges. Through the access switches and distribution switches, LAN clients connect to all back-end servers/storage, to one another, and to the WAN.



The sample network has redundant ARXes connected at each of the distribution switches. Physically, this is a one-armed connection; conceptually, the ARX has front-end clients in front of it and back-end servers and filers behind it. To simplify the examples, the access switches are removed from the remaining illustrations in the book.

The network filers all live on a class-C subnet at 192.168.25.x. These filers are called *back-end* filers, since they are the storage behind the *front-end* services of the ARX. The filers can be heterogeneous: NAS devices and file servers (possibly with additional DAS) need only support CIFS or NFS to be on the back end of the ARX.



Contacting Customer Service

You can use the following methods to contact F5 Networks Customer Service:

F5 Networks Online Knowledge Base Online repository of answers to frequently-asked questions.	http://support.f5.com
F5 Networks Services Support Online Online customer support request system	https://websupport.f5.com
Telephone	Follow this link for a list of Support numbers: http://www.f5.com/support/support-services/contact/

2

Managing Administrative Accounts

Overview

An administrative *user* is a login account for accessing the CLI or the GUI. One initial user account is created during the initial boot of the ARX; the initial-boot script prompts for this default account's username and password. This user account can effectively access all CLI commands and GUI screens.

The first section explains how to change a password for your user account. Most users can change their own passwords. The remaining sections are for administrators with a high security clearance; they involve managing administrative accounts and groups. The account that is created during installation has sufficient security clearance for all of these tasks.

Changing Your Password

You can change the password for the current administrative account at any time. We recommend that you do this from time to time for maximum security. From priv-exec mode, use the password command to change the account password:

```
password
```

This prompts for the current password, to verify that you are authorized to change the password. Then it asks for the new password twice, to confirm that it is correct. The new password can be 28 characters at most. For maximum security, create a password with at least one of each of the following:

- lower-case letter (a-z),
- upper-case letter (A-Z),
- number (0-9), and
- non alpha-numeric symbol (!, $, #, *, @, and so forth). The following characters are *not* allowed: | & ; ( ) < > ` -.

For example:

```
bstnA# password
Old Password: old_pass
New Password: sp1ffyN3wPa$$wd
Validate Password: sp1ffyN3wPa$$wd
bstnA# ...
```

Adding an Administrative User

You may want to manage your administration by adding additional accounts, perhaps without access to the entire CLI-command set. You can create accounts that adhere to certain administrative *roles* (such as a storage engineer or a network technician), and therefore have access only to commands that pertain to their role. You may also want to simplify SSH access to the accounts by installing public keys into them. If you do not require this level of control over your administrators, you can skip the rest of this chapter.

From gbl mode, use the user command to create a new user account:

user *username*

where ***username*** (1-32 characters) is a user name that you choose. As with the password, you cannot use any of the following characters: **| & ; ( ) < > ` -**.

The CLI prompts for the user's password, then for validation of the password. Choose your password with the same guidelines described above. This places you in gbl-user mode, where you have access to commands for editing the user account.

For example, the following command creates a new user, "newadmin:"

```
bstnA(gbl)# user newadmin
Password: 5up3r$ECRETpw
Validate Password: 5up3r$ECRETpw
bstnA(gbl-user[newadmin])# ...
```

Changing the User's Password

From gbl-user mode, you can use the password command to change the account password:

password

This invokes the same password prompts that appear when a new account is created: the CLI prompts for the user's password, then for validation of the password. Follow the password guidelines that were described earlier in the chapter (recall *Changing Your Password*, on page 2-3).

For example, the following command sequence changes the password for the user, "testadmin:"

```
bstnA(gbl)# user testadmin
bstnA(gbl-user[testadmin])# password
Password: an3Wpa$$wd
Validate Password: an3Wpa$$wd
bstnA(gbl-user[testadmin])# ...
```

Listing All Groups

The next section will explain how to add the current user to a *group*. A user account's group determines its access privileges. It is helpful to have a list of available groups before you begin; several are configured at the factory. Use the show group all command to display all configured groups:

```
show group all
```

For example:

```
bstnA(gbl-group[admins])# show group all

Configured Groups

     Group Name
     -------------
     operator
     network-technician
     storage-engineer
     network-engineer
     crypto-officer
     backup-operator

bstnA(gbl-group[admins])# ...
```

Adding the User to a Group

The next step in configuring a user is adding it to a group. By default, a new user belongs to an administrative *group* with access to a subset of CLI commands (primarily "show" commands); you can change this by adding the user account to a group with access to other commands. An administrative group is a set of users with the same administrative role.

As shown in the above section, there are several factory-set groups on the system:

- operator,
- network-technician,
- storage-engineer,
- network-engineer,
- crypto-officer, and
- backup-operator.

Each group is associated with one *role*, so members of each group can access the set of CLI commands for that role. The *ARX® CLI Reference* shows the role (or roles) that can access each command in the CLI. The *Security Table*, on page A-1 lists all CLI commands and the roles that can use them.

A new user belongs to the "operator" group by default, so it can access commands associated with monitoring the system. This group cannot change system configuration. From gbl-user mode, use the group command to choose an additional group for the user:

group *group-name*

where ***group-name*** (up to 64 characters) identifies a configured group.

A user can belong to multiple groups.

For example, the following command sequence adds the "newadmin" user to the "crypto-officer" group:

```
bstnA(gbl)# user newadmin
bstnA(gbl-user[newadmin])# show group all

Configured Groups

     Group Name
     -------------
     operator
     network-technician
     storage-engineer
     network-engineer
     crypto-officer
     backup-operator
     admins

bstnA(gbl-user[newadmin])# group crypto-officer
bstnA(gbl-user[newadmin])# ...
```

Listing all Group Users

Use the show group users command to show all configured users for all configured groups:

show group users

For example, this shows that the “newadmin” user belongs to two groups, “operator” and “crypto-officer:”

```
bstnA(gbl-group[admins])# show group users
Group Users
-----------

Group                          User
-----------------------------  -----------------------------
admins                         adm1
admins                         adm12
crypto-officer                 admin
crypto-officer                 newadmin
operator                       admin
operator                       newadmin
operator                       adm1
operator                       adm12

bstnA(gbl-group[admins])# ...
```

Removing the User from a Group

Use the no group command to remove the user from one group:

no group *group-name*

where ***group-name*** (up to 64 characters) is the group to remove.

Important

If the user does not belong to any group, the account does not allow any access to the CLI or the GUI.

For example, the following command sequence removes the “newadmin” user from the “network-technician” group:

```
bstnA(gbl)# user newadmin
bstnA(gbl-user[newadmin])# no group network-technician
bstnA(gbl-user[newadmin])# ...
```

Simplifying SSH Access to the Account (optional)

When an administrator logs into their user account through SSH, the ARX prompts for a password. To avoid this prompt and login immediately, the administrator can copy his *public key* from his remote management station to this account. To paste a public key into the current user account, use the ssh-key command from gbl-user mode:

ssh-key {dsa | rsa | rsa1} *"public-key"*

where

dsa | rsa | rsa1 chooses the client-side encryption algorithm and SSH protocol:

- **dsa** indicates DSA encryption for SSHv2,
- **rsa** indicates RSA encryption for SSHv2, and
- **rsa1** indicates RSA encryption for SSHv1. The ARX does not support SSHv1 by default; you can enter an rsa1 key here and enable SSHv1 later (as described in a later chapter).

public-key (1-2500 characters) is the administrator's public RSA or DSA key. This is usually a very long, cryptic string that you can access at the remote management station. Paste it into the CLI, surrounded by quotation marks ("").

Important

It is possible to copy from one window to another and have one of the windowing systems break up this long line with <Return> or <Line-Feed> characters. When you paste the string, verify that it is all on a single line.

A single user may enter the account from multiple management stations. Each combination of administrator and remote-management station has a unique public key. A user account therefore supports multiple keys. Re-use the ssh-key command once for each public key.

For example, the following command sequence outputs juser's public DSA key at a remote management station ("clientLinux," a Linux box), then goes to the ARX to paste juser's public key into the "newadmin" user account:

At the Remote Management Station:

```
juser@clientLinux:/$ cd ~/.ssh
juser@clientLinux:~/.ssh$ cat id_dsa.pub
ssh-dss
AAAAB3NzaC1kc3MAAACBAPqSVxs6Soxs5D9G7Ul8dQrf7Eo7vNdTawaH0K7DsyV2ND0RqxttRtNpw/fdIcm5cHOrYW4OYL6HJe
sMeJPguAzY8hbTkwsz+uRJLFnmRTy236DXDFiTc38Er6UQCoa1On9VrKWhoEGNe1YCn+cIsb3S+s44QPOx9GPFSVN1hqdVAAAA
FQC8x+2VKzUH16xrAMKuvVh50c53lwAAAIEAvOgRX8Ek2e/uCCJXlme0n7EsL3+yTEsOP7C9Bsl05KoCAgCSYP8G/1rc372Vy0
xF3PGL9QsI/bj+48SEAuJJTpJR1eB9MLpwmraVa/IsX16Xhr34eLDwH3NwtlwqRH9fhkjnWwhEoLRC7Bf/g493HoXPD2dNjbKv
qiMgq+s7CBEAAACAcAF+a+S/0OUNfpuv6QPV+SX9WoaazJthtUiP8pI4yl6sVAhp3Op5LxWT58Xl4ed+F0vUR2cfdjAF23YGYR
wK2c2h4FjnoBjLuoodhXJ+xAC/DPb4EvwEcBtqlPnpWzsPlAFX/I1pPA4fUyUOOifCrP12etsoZ9mnxawLRAAEa+A=
juser@clientLinux
juser@clientLinux:~/.ssh$ ...
```

At the ARX:

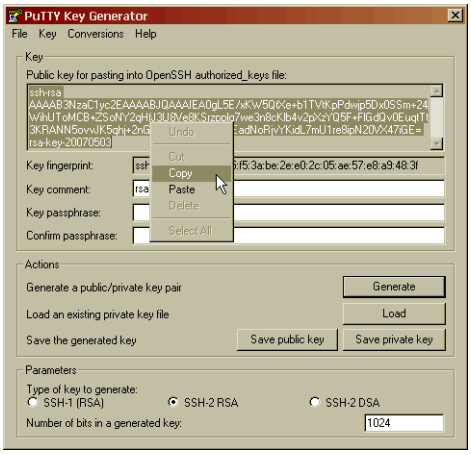
```
bstnA(gbl)# user newadmin
```

```
bstnA(gbl-user[newadmin])# ssh-key dsa “ssh-dss
AAAAB3NzaC1kc3MAAACBAPqSVxs6Soxs5D9G7Ul8dQrf7Eo7vNdTawaH0K7DsyV2ND0RqxttRtNpw/fdIcm5cHOrYW4OYL6HJe
sMeJPguAzY8hbTkwsz+uRJLFnmRTy236DXDFiTc38Er6UQCoa1On9VrKWhoEGNe1YCn+cIsb3S+s44QPOx9GPFSVN1hqdVAAAA
FQC8x+2VKzUH16xrAMKuvVh50c53lwAAAIEAvOgRX8Ek2e/uCCJXlme0n7EsL3+yTEsOP7C9Bsl05KoCAgCSYP8G/1rc372Vy0
xF3PGL9QsI/bj+48SEAuJJTpJR1eB9MLpwmraVa/IsX16Xhr34eLDwH3NwtlwqRH9fhkjnWwhEoLRC7Bf/g493HoXPD2dNjbKv
qiMgq+s7CBEAAACAcAF+a+S/0OUNfpuv6QPV+SX9WhoazJthtUiP8ts4yl6sVAhp3Op5LxWT58Xl4ed+F0vUR2cfdjAF23YGYR
wK2c2h4FjnoBjLuoodhXJ+xAC/DPb4EvwEcBtqlPnpWzsPlAFX/I1pPA4fUyUOOifCrP12etsoZ9mnxawLRAAEa+A=
juser@clientLinux”
bstnA(gbl-user[newadmin])# ...
```

PuTTYgen Example

As another example, this command sequence uses the PuTTYgen application (freely available on the Internet) to generate an RSA key, then copies the key and pastes it into the CLI:

At the Remote Management Station:



At the ARX:

```
bstnA(gbl)# user testAccount
bstnA(gbl-user[testAccount])# ssh-key rsa “ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIEA0gL5E/xKW5QtXe+b1TVtKpPdwjp5Dx0SSm+24WihUToMCB+ZSoNY2qHfJ3U8Ve8KSrzppl
g7we3n8cKlb4v2pXzYQ5F+FlGdQv0EuqtTt3KRANN5ovvJK5qhj+2nGFSbJNfjVy1O5PNEadNoRjvYKidL7mU1re8ipN20VX47
iGE= rsa-key-20070503”
bstnA(gbl-user[testAccount])# ...
```

Showing the Public Keys in All User Accounts

Use the show ssh-user command to view all user accounts with SSH keys:

```
show ssh-user
```

For example,

```
bstnA(gbl-user[newadmin])# show ssh-user

User            KeyId      Type  Fingerprint
--------------- ---------- ----- -----------------------------------------------
newadmin        1          dsa   1e:ca:4b:9f:0d:51:97:98:1f:d8:26:81:8e:ab:3d:9b
adm12           2          dsa   8d:45:67:07:53:5f:61:9c:54:47:a8:76:4c:9a:93:05
admin           3          dsa   0b:fa:85:93:15:f9:ff:7e:8a:4c:1b:16:ba:4a:de:e7
```

```
admin           4          dsa   1e:ca:4b:9f:0d:51:97:98:1f:d8:26:81:8e:ab:3d:9b
bstnA(gbl-user[newadmin])# ...
```

This displays the fingerprint for each public key. The fingerprint uniquely identifies the public key, it is much shorter, and it is displayed by many client machines instead of the full key. For example, this shows the fingerprint for "juser@clientLinux" on a Linux management station:

```
juser@clientLinux:~/.ssh$ ssh-keygen -l -f id_dsa.pub
1024 1e:ca:4b:9f:0d:51:97:98:1f:d8:26:81:8e:ab:3d:9b id_dsa.pub
juser@clientLinux:~/.ssh$ ...
```

Focusing on One Account

To see the public keys for one account only, add the username to the end of the command:

show ssh-user *username*

where ***username*** (1-32 characters) identifies the administrative-user account to display.

For example, this shows the only public-key fingerprint configured for the "newadmin" account:

```
bstnA(gbl-user[newadmin])# show ssh-user newadmin

User            KeyId      Type  Fingerprint
--------------- ---------- ----- -----------------------------------------------
newadmin        1          dsa   1e:ca:4b:9f:0d:51:97:98:1f:d8:26:81:8e:ab:3d:9b
bstnA(gbl-user[newadmin])# ...
```

Removing a Public Key from the Account

You can use the KeyId or the Fingerprint from the show-command output, above, to remove a key from a user account. From gbl-user mode, use the no ssh-key command to remove a public SSH key:

no ssh-key {fingerprint *fingerprint* | id *key-id*}

where you can choose between two possible identifiers:

fingerprint (1-50 characters) is the full fingerprint, from the show ssh-user output, or

key-id (1-2,147,483,647) is an integer to identify the fingerprint/public key. This also appears in the show ssh-user output, in the KeyId column.

For example, this command sequence shows all public keys and deletes one of them based on its fingerprint:

```
bstnA(gbl)# user admin
bstnA(gbl-user[admin])# show ssh-user admin

User            KeyId      Type  Fingerprint
--------------- ---------- ----- -----------------------------------------------
admin           3          dsa   0b:fa:85:93:15:f9:ff:7e:8a:4c:1b:16:ba:4a:de:e7
admin           4          dsa   1e:ca:4b:9f:0d:51:97:98:1f:d8:26:81:8e:ab:3d:9b
bstnA(gbl-user[admin])# no ssh-key fingerprint 0b:fa:85:93:15:f9:ff:7e:8a:4c:1b:16:ba:4a:de:e7
bstnA(gbl-user[admin])# ...
```

As another example, this removes a public key based on its ID:

```
bstnA(gbl)# user adm12
bstnA(gbl-user[adm12])# show ssh-user adm12

User            KeyId      Type  Fingerprint
--------------- ---------- ----- -----------------------------------------------
adm12           2          dsa   8d:45:67:07:53:5f:61:9c:54:47:a8:76:4c:9a:93:05
bstnA(gbl-user[adm12])# no ssh-key id 2
bstnA(gbl-user[adm12])# ...
```

Removing All Keys of a Given Type

You can use the no ssh-key command to remove all DSA/SSHv2, RSA/SSHv2, or RSA/SSHv1 keys from the current account:

```
no ssh-key {dsa | rsa | rsa1}
```

where **dsa** | **rsa** | **rsa1** chooses the type of key to remove.

This applies to the current account only.

For example, this command sequence removes all RSA/SSHv1 keys from the “admin” account:

```
bstnA(gbl)# user admin
bstnA(gbl-user[admin])# no ssh-key rsa1
bstnA(gbl-user[admin])# ...
```

Removing All Keys from the Account

You can use the no ssh-key all command to remove all keys at once from the current account:

```
no ssh-key all
```

As above, this applies to the current account only.

For example, this command sequence removes all public SSH keys from the “adm12” account:

```
bstnA(gbl)# user adm12
bstnA(gbl-user[adm12])# no ssh-key all
bstnA(gbl-user[adm12])# ...
```

Adding a Group (optional)

For most installations, the default groups should suffice. You can optionally use the gbl-mode group command to add a new group:

`group` ***`name`***

where ***name*** (up to 64 characters) is the name you choose for the new group.

This places you into gbl-group mode, where you must set the group *role*. The role determines the group's set of CLI commands. From gbl-group mode, you can also add users to (or delete users from) a group.

For example, the following command creates a new group, "admins:"

```
bstnA(gbl)# group admins
bstnA(gbl-group[admins])# ...
```

Setting the Group Role

The next step in configuring a new group is to provide the group with one or more roles; factory-default groups already have roles. The *role* determines the set of CLI commands for the group's members. The following roles are available:

- **operator** is a clerical administrator,
- **network-technician** configures layer-2 and IP networks under the guidance of a network-engineer,
- **network-engineer** designs network topologies,
- **storage-engineer** designs and configures network storage,
- **crypto-officer** keeps passwords and manages network security, and
- **backup-operator** runs a small set of commands to backup and restore files in a namespace volume (described in the *ARX® CLI Storage-Management Guide*).

Each role has access to its own set of relevant commands, leaving out any commands that do not apply. For example, the operator can access most of the "show" commands but no commands that can change system configuration. The network technician can access commands that change the layer-2 and layer-3 configuration. The storage engineer can access commands that manage storage on the switch.

Any CLI command may be accessible by more than one role; many commands, for example, are accessible by both the network-technician and network-engineer roles.

The crypto-officer, ostensibly, only accesses the commands that are relevant to system security. This makes security easier to manage by reducing the number of visible commands in the CLI. The crypto-officer is also the only role that can access the commands in this chapter. If there is need for a

crypto-officer to access a command that is otherwise inaccessible, he can gain access by adding his user account to another group (as described above) or editing the role of his or her group (shown below).

CLI Security Levels, an appendix, contains a table of all CLI commands and the Security Role(s) to run them.

◆ Note

An administrative account with the "crypto-officer" role can log into an ARX soon after a reboot, before its database is fully accessible. Administrative accounts with lesser roles must wait until the database is fully configured before a login is possible.

From gbl-group mode, use the role command to set the group role:

```
role {operator | backup-operator | network-technician |
network-engineer | storage-engineer | crypto-officer}
```

where **operator | ... crypto-officer** is a required choice.

You can use this command multiple times to assign multiple roles to the group.

For example, the following command sequence provides two roles for the new group, "admins:"

```
bstnA(gbl)# group admins
bstnA(gbl-group[admins])# role storage-engineer
bstnA(gbl-group[admins])# role network-engineer
bstnA(gbl-group[admins])# ...
```

Showing all Group Roles

Use the show group roles command to show all configured roles for all configured groups:

```
show group roles
```

For example:

```
bstnA(gbl-group[admins])# show group roles
Group Roles
-----------

Group                          Roles
-----------------------------  -----------------------------
admins                         storage-engineer
backup-operator                backup-operator
crypto-officer                 crypto-officer
network-engineer               network-engineer
network-technician             network-technician
operator                       operator
storage-engineer               storage-engineer

bstnA(gbl-group[admins])# ...
```

Removing a Group Role

From gbl-group mode, use the no role command to remove one group role:

```
no role {operator | backup-operator | network-technician |
network-engineer | storage-engineer | crypto-officer}
```

where **operator | ... crypto-officer** is a required choice.

Important

If a group has no role configured, none of the group's users can access the CLI or GUI.

For example, the following command sequence removes the "storage-engineer" role and adds "crypto-officer" for the group, "admins:"

```
bstnA(gbl)# group admins
bstnA(gbl-group[admins])# no role storage-engineer
bstnA(gbl-group[admins])# role crypto-officer
bstnA(gbl-group[admins])# ...
```

Listing All Administrative Users

The next step in configuring a group will be to add users to it; it is useful to have a list of configured users when you do this. To access a list of all configured users, use the show users command:

```
show users
```

This also shows the groups for the current user account.

For example:

```
bstnA(gbl-group[admins])# show users

Users
    Configured Users
    ----------------
    adm1
    adm12
    admin
    newadmin


    Current User
    ------------
    admin
        group   operator
        group   crypto-officer

bstnA(gbl-group[admins])# ...
```

Adding a User to the Group

The final step in configuring a group is to add users to it. Every user must belong to at least one group for the account to function. From gbl-group mode, use the user command to add a user to the current group:

```
user username
```

where ***username*** (up to 64 characters) identifies a configured user. Use the show users command for a list of all configured users.

For example, the following command sequence adds the “adm1” user to the new group, “admins:”

```
bstnA(gbl)# group admins
bstnA(gbl-group[admins])# show users

Users
    Configured Users
    ----------------
    adm1
    adm12
    admin
    newadmin


    Current User
    ------------
    admin
        group   operator
        group   crypto-officer

bstnA(gbl-group[admins])# user adm1
bstnA(gbl-group[admins])# ...
```

Removing a User from the Group

From gbl-group mode, use the no user command to remove a user from the current group:

no user *username*

where ***username*** (up to 64 characters) identifies a configured user. Use the show group users command for a list of all users in all groups.

For example, the following command sequence removes the “adm12” user from the group, “admins:”

```
bstnA(gbl)# group admins
bstnA(gbl-group[admins])# show group users
Group Users
-----------

Group                          User
-----------------------------  -----------------------------
admins                         adm1
admins                         adm12
crypto-officer                 admin
crypto-officer                 newadmin
operator                       admin
operator                       newadmin
operator                       adm1
operator                       adm12

bstnA(gbl-group[admins])# no user adm12
bstnA(gbl-group[admins])# show group users
Group Users
-----------

Group                          User
-----------------------------  -----------------------------
admins                         adm1
crypto-officer                 admin
crypto-officer                 newadmin
```

```
operator                        admin
operator                        newadmin
operator                        adm1

bstnA(gbl-group[admins])# ...
```

Removing an Administrative User

From gbl mode, use the no user command to remove the configuration for an administrative-user account:

no user *username*

where ***username*** (1-32 characters) identifies the user account to remove.

For example, this removes an account named “guestUser:”

```
bstnA(gbl)# no user guestUser
bstnA(gbl)# ...
```

3

Configuring Layer 2

Overview

The ARX's default behavior is identical to that of a NAS filer; it does not forward any layer-2 packets from one port to another. For some installations, the default configuration for layer-2 may be sufficient; read the first several sections of this chapter to learn all the defaults and verify that they are adequate. You can enable at least one interface (described later in the chapter) and continue to the next chapter if the defaults are sufficient.

The ARX can optionally function as a Media Access Control (MAC) bridge conforming to the IEEE 802.1D specification. Layer-2 bridging is typically not required for most installations and is therefore disabled at installation time. For an installation that requires these features, use the remaining sections to configure additional VLANs, packet forwarding (typically with spanning-tree), link aggregation, and/or other layer-2 options.

Special Considerations For Layer 2 Configuration On Newer ARX Platforms

Changes in the designs of newer ARX platforms require that some aspects of their layer 2 networking configuration differ from the common configuration procedures described here for all ARX platforms. This section notes the differences in the new platform designs that cause their layer 2 configurations to be different; the actual model-specific exceptions to the common configuration procedures will be noted with the procedures themselves.

Special Considerations For ARX-1500 and ARX-2500

Note the following considerations when using the ARX-1500 and ARX-2500:

- The ARX-1500 and ARX-2500 do not support layer 2 switching.
- The ARX-1500 and ARX-2500 do not provide dedicated management interfaces.
- The ARX-1500 and ARX-2500 have only the default VLAN, that is, VLAN 1, configured by default. They do not have either a private VLAN (VLAN 1002, configured as tagged) or a metalog VLAN (VLAN 1003, configured as tagged) configured in their factory default configuration, as is the case with earlier ARX models.

Special Considerations For ARX-VE

Note the following considerations when using the ARX-VE:

- The ARX-VE is a software-only virtual appliance, and, as such, functionality related to physical network infrastructure is not relevant to its operation. This includes VLANs, forwarding, and the use of out-of-band management interfaces.
- In some cases, networking functionality is provided on the ARX-VE's behalf by VMware. This includes Ethernet interfaces and 802.3ad link aggregation, as well as high-availability redundancy.

Concepts and Terminology

The ARX ships with three default VLANs configured. These are required for proper operation:

- VLAN 1 (untagged), by default, has all client/server ports as its members.
- The *Private VLAN* is reserved for a private network that connects the ARX's internal processors to one another. This is intended to be entirely distinct from any VLAN on a connected LAN. This is VLAN 1002 (tagged) by default. The ARX-1500 and ARX-2500 do not have a private VLAN configured in their factory default configuration.
- The *Metalog VLAN* is reserved for a metalog network. Internal processors use this network to send critical log information for safe storage. This is also intended to be distinct from external VLANs, as well as from the private VLAN. This is VLAN 1003 (tagged) by default. The ARX-1500 and ARX-2500 do not have a metalog VLAN configured in their factory default configuration.

If you join multiple ARXes into redundant pairs and/or a larger ARX network (all described in later chapters), they communicate with one another over their respective private or metalog VLANs. The private and metalog VLANs must be the same for all such switches. In the cases of the ARX-1500 and ARX-2500, a VLAN must be configured explicitly to support the redundancy connection.

Default Layer-2 Configuration

- As stated above, three default VLANs are configured:
  - VLAN 1 (untagged) is the default for all client/server traffic.
  - 1002 (tagged) is the default for the private VLAN. The ARX-1500 and ARX-2500 do not have this configured by default.
  - 1003 (tagged) is the default for the metalog VLAN. The ARX-1500 and ARX-2500 do not have this configured by default.
- Switch forwarding is disabled; a packet coming into one port is never forwarded to another. The switch behaves as an end station in the local spanning tree. The ARX-1500 and ARX-2500 do not support switch forwarding.
- All ports (called *interfaces* in the CLI) are shut down, with
  - speed set to "auto-negotiate,"
  - flow control disabled, and
  - frame size (MTU) of 1500 bytes.
- No channels (as defined by IEEE 802.3ad, Link Aggregation)
- The ARX-500 has a port that is already designated as a redundancy link, port 1/2. You cannot change this.
- The ARX-1500 and ARX-2500 do not provide a dedicated out-of-band management interface; for these models, port 1/1 is configured by default as the management interface.

Refer to the following sections if you want to add VLANs, allow switch forwarding (and possibly change the default spanning tree settings), change the interfaces, configure channels, or configure redundancy ports.

Adding a VLAN

A Virtual Bridged LAN (VLAN), defined in IEEE 802.1Q, is a logical grouping of MAC addresses. Devices on the same VLAN appear to be located on the same LAN segment, even though some of the devices may be on different floors or different buildings. The logical grouping for a VLAN is often based on department or division membership in a company.

From cfg mode, use the vlan command to add a VLAN to the ARX:

vlan *id*

where ***id*** is the integer ID (1-4095) for the VLAN.

This puts you into cfg-vlan mode, where you identify the ports that belong to the VLAN.

For example, the following command sequence creates an empty configuration for VLAN 25:

```
bstnA(cfg)# vlan 25
bstnA(cfg-vlan[25])# ...
```

A VLAN cannot have more than one member interface or channel enabled.

Adding Ports to the VLAN

The ARX-500 has a single client/server port, 1/1, which is automatically a tagged member of any VLAN you configure. Skip to the next section (*Setting a VLAN Description (optional)*, on page 3-8) if you are setting up an ARX-500 chassis.

For platforms with multiple ports, the next step in configuring a VLAN is to identify the ports that belong to it. This determines the ports that the ARX uses to send packets out to the VLAN. For example, a broadcast to this VLAN goes out over all the ports that you configure here. A port can be a member of more than one VLAN if the port has VLAN tagging enabled; VLAN tagging is described later in this chapter.

From cfg-vlan mode, use the members command to define a range of physical ports that belong to the current VLAN:

members *slot/port*

where ***slot/port*** identifies a single port, or

members *slot/port* to *slot/port*

where ***slot/port* to *slot/port*** defines a range of physical Ethernet ports; for example, “2/3 to 2/14” (on an ARX-4000), or “1/2 to 1/4” (on an ARX-2000). Use the show interface summary command to find all ports on the ARX.

You can use the members command multiple times to define more than one range of ports.

For example, the following command sequence sets two port ranges for VLAN 25:

```
bstnA(cfg)# vlan 25
```

```
bstnA(cfg-vlan[25])# members 2/3 to 2/4
bstnA(cfg-vlan[25])# members 2/6 to 2/7
bstnA(cfg-vlan[25])# ...
```

As another example, the following command sequence adds port 2/11 to VLAN 4:

```
bstnA(cfg)# vlan 4
bstnA(cfg-vlan[4])# members 2/11
bstnA(cfg-vlan[4])# ...
```

Adding Ports with 802.1Q-VLAN Tagging Enabled

VLAN *tagging* allows more than one VLAN to exist on a single port; tagging must be enabled for a port to be a member of multiple VLANs. A port with tagging enabled adds the appropriate 802.1Q VLAN ID (VID) to an Ethernet frame before relaying the frame to a LAN segment. This applies to outbound (egress) frames only. Typically, VLAN tagging is used to transport multiple VLANs between switches or to an end station that supports multiple VLANs.

Devices that support 802.1Q read the tag to determine the destination VLAN for the frame. Older devices that do not support 802.1Q may reject a tagged Ethernet frame as invalid.

Ports added by the members command have tagging disabled by default. You can set tagging on current member ports, or you can add new ports to the VLAN with tagging enabled. From cfg-vlan mode, use the tag command to add (or re-configure) ports with tagging enabled:

tag ***slot/port*** **[to** ***slot/port*****]**

where ***slot/port*** **to** ***slot/port*** defines a range of physical Ethernet ports (for example, “2/3 to 2/6”).

You can use the tag command multiple times to define more than one range of ports.

For example, the following command sequence adds a range of tag-enabled ports for VLAN 5. After these commands, ports 2/3 through 2/6 will tag all VLAN 5 frames:

```
bstnA(cfg)# vlan 5
bstnA(cfg-vlan[5])# tag 2/3 to 2/6
bstnA(cfg-vlan[5])# ...
```

Disabling Tagging on Member Ports

Tagging is disabled by default. From cfg-vlan mode, use the no tag command to disable tagging on a member port:

no tag ***slot/port*** **[to** ***slot/port*****]**

where ***slot/port*** **to** ***slot/port*** defines a range of physical Ethernet ports (for example, “2/9 to 2/12”).

For example, the following command sequence disables tagging for VLAN 5 on port 2/4:

```
bstnA(cfg)# vlan 5
bstnA(cfg-vlan[5])# no tag 2/4
bstnA(cfg-vlan[5])# ...
```

Removing Ports from a VLAN

Use the no members command to remove a range of ports from the current VLAN:

no members ***slot/port*** **[to** ***slot/port*****]**

where ***slot/port*** **to** ***slot/port*** defines a range of physical Ethernet ports (for example, “2/5 to 2/8”).

For example, the following command sequence sets a large port range for VLAN 25, then removes two ports from the middle of the range:

```
bstnA(cfg)# vlan 25
bstnA(cfg-vlan[25])# members 2/3 to 2/10
bstnA(cfg-vlan[25])# no members 2/5 to 2/6
bstnA(cfg-vlan[25])# ...
```

Setting a VLAN Description (optional)

A VLAN description is an optional label that appears in the show vlan output. From cfg-vlan mode, use the description command to add a description to the current VLAN:

description ***vlan-description***

where ***vlan-description*** (up to 80 characters) is a description that you choose for the VLAN. Quote the string if it contains any spaces.

For example, the following command sequence assigns the description, “personnel dept.,” to VLAN 25:

```
bstnA(cfg)# vlan 25
bstnA(cfg-vlan[25])# description “personnel dept.”
bstnA(cfg-vlan[25])# ...
```

Removing the VLAN Description

From cfg-vlan mode, use no description to remove the VLAN description:

no description

For example:

```
bstnA(cfg)# vlan 25
bstnA(cfg-vlan[25])# no description
bstnA(cfg-vlan[25])# ...
```

Enabling Jumbo Frames (optional)

The default MTU for VLAN ports is 1500 bytes. From cfg-vlan mode, you can use the jumbo mtu command to use jumbo-size frames instead:

jumbo mtu ***bytes***

where ***bytes*** (1530-9198) is the packet size for this VLAN.

All member ports send and expect this frame size when communicating over the VLAN.

For example:

```
bstnA(cfg)# vlan 25
bstnA(cfg-vlan[25])# jumbo mtu 9000
bstnA(cfg-vlan[25])# ...
```

Important

All interfaces accept jumbo frames and most record them as "Good Oversize Frames" whether the VLAN is configured for them or not; the ARX-1500 and ARX-2500 do not count these statistics, however. (The interface statistics that show this count are described below: see Showing Interface Statistics, on page 3-29.) If the frame is to be switched at layer 2, the layer-2 software successfully forwards the frame to its destination no matter what size is set for the VLAN. However, these oversize frames are discarded by higher-level processes such as NFS or CIFS.

Important

For communication to succeed, all devices in the VLAN must be configured consistently for jumbo or standard-size frames.

Reverting to the Standard-Frame Size

From cfg-vlan mode, use the no jumbo mtu command to revert to the default frame size, 1500 bytes:

```
no jumbo mtu
```

For example:

```
bstnA(cfg)# vlan 3
bstnA(cfg-vlan[3])# no jumbo mtu
bstnA(cfg-vlan[3])# ...
```

Listing all VLANs

Use the show vlan summary command to list all configured VLANs:

```
show vlan summary
```

For example:

```
bstnA(cfg)# show vlan summary

Total VLANs Configured: 4

VLAN ID      Usage     Channel      Frame/MTU     Description
-------------------------------------------------------------
1           External    N/A         Standard      Default VLAN.
25          External    N/A         Standard      personnel dept.
1010        Private     N/A         Standard      Private Subnet VLAN.
1011        Metalog     N/A         Standard      Private Metalog VLAN.


bstnA(cfg)# ...
```

Showing Details For One VLAN

You can show all member ports, non-member ports, and tagged ports configured for a given VLAN. Use the show vlan command to see one VLAN's configuration:

show vlan *vlan-id*

where ***vlan-id*** (1-4096) identifies the VLAN.

For example, the following command shows VLAN 1:

```
bstnA(cfg)# show vlan 1

Vlan Id     : 1
Description : Default VLAN.
Frame/MTU   : Standard

Members  Non-Members   Tag
         (Blocked)
  S/P       S/P        S/P
-----------------------------
2/1       N/A         N/A
2/2       N/A         N/A
2/3       N/A         N/A
2/4       N/A         N/A
2/5       N/A         N/A
2/6       N/A         N/A
2/7       N/A         N/A
2/8       N/A         N/A
2/9       N/A         N/A
2/10      N/A         N/A
2/11      N/A         N/A
2/12      N/A         N/A
2/13      N/A         N/A
2/14      N/A         N/A

bstnA(cfg)#
```

Removing a VLAN

Removing a VLAN disables all port-configuration parameters related to that VLAN. Its member ports, if not tagged for any other VLAN, default to VLAN 1. Layer-3 objects associated with the VLAN, such as IP addresses, are disabled. You cannot remove VLAN 1, the default VLAN on the ARX. From cfg mode, use the no form of the vlan command to remove a VLAN configuration:

no vlan *id*

where ***id*** (1-4095) identifies the VLAN to remove.

For example, the following command sequence deletes the configuration for VLAN 11:

```
bstnA(cfg)# no vlan 11
bstnA(cfg)# ...
```

Changing the Private VLAN

The private VLAN is internal to the switch, and cannot intersect with an external VLAN. It is used for internal communication and private communication with other ARXes, such as a redundant peer. The default number for the private VLAN is 1002. The ARX-1500 and ARX-2500 do not have a private VLAN configured by default.

From cfg mode, you can use the ip private vlan command to reconfigure the private VLAN number:

```
ip private vlan internal vlan-id
```

where ***vlan-id*** (1-4095) is the new VLAN. You must enter a VLAN that has been configured on the ARX: use the show vlan summary command for a list of configured VLANs (see *Listing all VLANs*, on page 3-9).

Important

Any change to the private VLAN causes the switch to reboot. Consult F5 technical support before changing this address.

The CLI prompts you before changing the private VLAN and rebooting. Enter **yes** to proceed.

For example:

```
bstnA(cfg)# ip private vlan internal 1008
Change the private VLAN and reboot the chassis? [yes/no] yes
Broadcast message from root (console) (Thu Apr  1 18:05:15 2004):

The system is going down for reboot NOW!
...
```

Changing the Metalog VLAN

The *metalog* VLAN is another private VLAN, used to record real-time file-system transactions on persistent storage. Specifically, ASM processors send namespace-log data over the metalog to the SCM. The ASM can use these logs for failure recovery. Like the private VLAN, the metalog VLAN is defined in the switch's initial-boot script; if this VLAN is already used in your network, you must change the metalog VLAN along with the private VLAN. The metalog-VLAN number is 1003 by default. The ARX-1500 and ARX-2500 do not have a metalog VLAN configured by default; for these models, metalog data is exchanged via a layer-3 connection, instead.

Use the metalog clause with the ip private vlan command to change the VLAN for the metalog:

```
ip private vlan internal vlan-id metalog metalog-vlan
```

where

vlan *id* is described above, and

metalog *metalog-vlan* (1-4095) selects a VLAN for metalog multicasts. If you omit this clause, as shown in the previous sections, the metalog VLAN remains unchanged. You must enter a VLAN that has been configured on the ARX: use the show vlan summary command for a list of configured VLANs (see *Listing all VLANs*, on page 3-9).

Important

As stated above, any change to the private VLANs causes a reboot. Consult F5 technical support before changing this address.

For example:

```
bstnA(cfg)# ip private vlan internal 1010 metalog 1011
Change private vlan and reboot the system? [yes/no] yes
Broadcast message from root (console) (Thu Apr  1 18:05:15 2004):

The system is going down for reboot NOW!
...
```

Changing the Private Subnet

The private subnet is used for inter-process communication over the private VLAN. You can change it whenever you change the private VLAN:

ip private vlan internal *vlan-id* [metalog *metalog-vlan*] subnet *net mask*

where

vlan-id and **[metalog *metalog-vlan*]** are described above, and

net mask defines the private subnet (for example, 192.168.111.0 255.255.255.0). For the ARX-4000, the subnet must be at least a Class C network (255.255.255.0; 24 bits or less). For the smaller platforms (ARX-2000 and ARX-500) you can use a smaller subnet, with a mask of 26 bits (255.255.255.192) or less.

Important

As stated above, any change to the private VLANs causes a reboot. Consult F5 technical support before changing this address.

For example:

```
bstnA(cfg)# ip private vlan internal 1010 subnet 172.16.55.0 255.255.255.0
Change the private VLAN and reboot the chassis? [yes/no] yes
Broadcast message from root (console) (Thu Apr  1 18:05:15 2004):

The system is going down for reboot NOW!
...
```

Allowing the ARX to Forward Packets

By default, the ARX does not forward any packets from one client/server port to another. It does not behave as a MAC bridge, and participates in the local spanning tree only as an end station. In this mode, the ARX appears as a file server to clients, and as a client to the back-end filers. You can optionally enable layer-2 forwarding and spanning tree, thereby using the ARX as a MAC bridge in your network.

The ARX-500 has a single client/server port, 1/1, which is treated as an end station in the layer 2 network. It does not support bridging or spanning tree. Skip to the next section (*Configuring a Layer-2 Interface*, on page 3-23) if you are setting up an ARX-500 chassis.

The ARX-1500 and ARX-2500 do not support layer-2 forwarding.

From cfg mode, use the switch-forwarding enable command to use the packet-forwarding capabilities of the ARX:

```
switch-forwarding enable
```

The CLI prompts for confirmation before enabling switch forwarding. Enter **yes** to proceed.

For example:

```
bstnA(cfg)# switch-forwarding enable
Warning: Enable switch forwarding between all Ethernet Interfaces?
This will also enable the Spanning Tree Protocol. [yes/no] yes
bstnA(cfg)# ...
```

Spanning Tree Defaults

The ARX-1500 and ARX-2500 do not support spanning-tree processing.

As stated in the CLI warning, this activates spanning-tree processing. The ARX uses spanning tree as a safeguard against network loops. Spanning tree has the following defaults:

- Rapid Spanning Tree (RST),
- Bridge Priority of 61,440,
- Hello Time of 2 seconds,
- Max Age of 20 seconds,
- Forward Delay of 15 seconds, and
- each port set as follows:
  - *not* an RST Edge Port,
  - Port Cost of 20,000, and
  - Port Priority of 128.

If these defaults are sufficient, you can skip to the next major section, *Configuring a Layer-2 Interface*, on page 3-23. Otherwise, the subsections below explain how to modify the spanning-tree configuration.

Editing the Spanning Tree Parameters

When switch-forwarding is enabled, spanning tree is enabled by default. A *spanning tree*, defined in IEEE 802.1D, is a layer-2 topology designed to prevent loops in bridged networks. The Spanning-Tree Protocol (STP) identifies a root bridge in the network, and each LAN segment identifies a single bridge port that is closest to the root. Alternate bridge ports are blocked. This ensures that there is exactly one active pathway from one LAN segment to another, eliminating bridge loops. The spanning-tree protocol also has mechanisms for activating alternate bridges/ports for situations where equipment fails.

The ARX supports IEEE 802.1D, which defines spanning tree, as well as 802.1w, Rapid Reconfiguration of Spanning Tree.

A single spanning-tree instance is shared amongst all VLANs. From cfg mode, use the spanning-tree command to configure spanning tree:

```
spanning-tree
```

This puts you into cfg-stp mode, where you set the spanning-tree protocol and various configuration parameters.

For example:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# ...
```

Choosing the Protocol

The next step in configuring a spanning tree is to choose the spanning-tree protocol. The ARX supports two spanning-tree standards: the original spanning-tree protocol defined in IEEE 802.1D, and Rapid Spanning Tree (RST) from 802.1w. RST improves on the original implementation by more-rapidly converging on a new spanning-tree topology after a bridge or port failure. Conforming bridges (including the ARX) agree on the most-modern supported version of the protocol and use that version for all spanning-tree communication. You can choose a preferred protocol, presumably the same one configured for neighboring bridges.

The default protocol is RST. From cfg-stp mode, use the protocol command to choose the preferred spanning-tree protocol:

```
protocol {dot1d | rst}
```

where **{dot1d | rst}** is a required choice:

dot1d limits the ARX to the original STP (IEEE 802.1D). This rejects RST BPDUs.

rst runs RST from IEEE 802.1w, but is compatible with bridges that run the original STP (IEEE 802.1D).

For example, the following command sequence sets the spanning tree to use the older 802.1D standard:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# protocol dot1d
bstnA(cfg-stp)# ...
```

Setting the Bridge Priority (optional)

The next step in configuring a spanning tree is to set the ARX's bridge priority within the current spanning tree. The spanning-tree protocol compares bridge priorities to choose a root bridge. A lower number represents a higher priority. The default priority for standard 802.1 bridges is 32,768; if all bridges in a spanning tree have this default, they elect the bridge with the lowest MAC address.

The default bridge priority is 61,440, to avoid making the ARX into the spanning-tree root. From cfg-stp mode, use the priority command to set the bridge priority for the ARX:

priority *value*

where ***value*** (0-61440) is the bridge priority. Lower numbers are preferred in the root-election process. Use a multiple of 4096 (such as 0, 4096, 8192, or 12288).

For example, the following command sequence sets a bridge priority of 0 (zero) for the ARX. This would likely make the ARX the root of the spanning tree:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# priority 0
bstnA(cfg-stp)# ...
```

Returning to the Default Priority

The default bridge priority is 61,440. For most installations, this bridge priority would prevent the election of the ARX as the root. From cfg-stp mode, use no priority to revert to the default bridge priority:

no priority

For example, the following command sequence sets the default priority for the ARX:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# no priority
bstnA(cfg-stp)# ...
```

Setting the Hello Time (optional)

The next step in configuring the spanning tree is to set the Hello Time. The *Hello Time* is the interval (in seconds) between broadcasts of Bridge Protocol Data Units (BPDUs) to neighboring bridges. The BPDUs have spanning-tree topology information that a bridge uses to determine its role in the spanning tree.

The default Hello Time is 2 (seconds). From cfg-stp mode, use the hello-time command to set the Hello Time interval:

hello-time *value*

where ***value*** (1-10) is number of seconds between BPDUs.

For example, the following command sequence sets a Hello Time of 4 (seconds):

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# hello-time 4
```

```
bstnA(cfg-stp)# ...
```

Reverting to the Default Hello Time

The default Hello Time is 2 (seconds). From cfg-stp mode, use no hello-time revert to this default:

```
no hello-time
```

For example:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# no hello-time
bstnA(cfg-stp)# ...
```

Setting the Max Age (optional)

The next step in configuring the spanning tree is to set the Max Age. The *Max Age* is the time (in seconds) to keep BPDU information from a neighboring bridge port before declaring the port information "stale." If the Max Age is reached for a port, it is considered disconnected and the bridges begin converging on a new spanning-tree topology. The Max Age is typically three times the Hello Time; it should *always* be at least 2 * (Hello Time + 1 second).

The default Max Age is 20 (seconds). From cfg-stp mode, use the max-age command to set the Max Age:

```
max-age value
```

where ***value*** (6-40) is number of seconds before aging out a BPDU.

For example, the following command sequence sets a Max Age of 10 (seconds):

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# max-age 10
bstnA(cfg-stp)# ...
```

Reverting to the Default Max Age

The default Max Age is 20 (seconds). From cfg-stp mode, use no max-age revert to this default:

```
no max-age
```

For example:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# no max-age
bstnA(cfg-stp)# ...
```

Setting the Forward Delay (optional)

The next step in configuring the spanning tree is to set the Forward Delay. The *Forward Delay* is the time for ports to stay in the *listen* and *learn* states, waiting for the best BPDU frame to reach the ARX. This time should be at least twice the maximum transit time for a BPDU to traverse the entire network. This allows the bridges enough time to establish a new spanning-tree topology in case of a bridge or link failure.

The default Forward Delay is 15 (seconds). From cfg-stp mode, use the forward-delay command to set the Forward Delay:

```
forward-delay seconds
```

where ***seconds*** (4-30) is the Forward Delay, in seconds.

For example, the following command sequence sets the Forward Delay to 20:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# forward-delay 20
bstnA(cfg-stp)# ...
```

Reverting to the Default Forward Delay

The default Forward Delay is 15 (seconds). From cfg-stp mode, use no forward-delay revert to this default:

```
no forward-delay
```

For example:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# no forward-delay
bstnA(cfg-stp)# ...
```

Reverting Back to Spanning-Tree Defaults

You can revert the spanning-tree protocol and all the global parameters to their default values. Use the no spanning-tree command:

```
no spanning-tree
```

This resets Bridge Priority, Hello Time, Max Age, and Forward Delay with a single command. For example:

```
bstnA(cfg)# no spanning-tree
bstnA(cfg)# ...
```

Setting Port Parameters (optional)

The next step in configuring spanning tree is to address port configuration. The primary parameters to consider are Port Cost and Port Priority. These two values are used by neighboring bridges to choose a designated port. The neighboring bridges use the *designated port* for relaying their frames toward the spanning-tree root. By default, all ports use the following settings:

- Port Cost - 20,000
- Port Priority - 128

A ten-gigabit interface (only offered on the ARX-4000 platform) always uses default port cost and priority.

These defaults are sufficient for a configuration where you do not prefer one port to another for relaying frames. You have the option to reset these parameters on a per-port basis.

Setting the Port Cost

The first step in configuring a port for spanning-tree is to address Port Cost. The Port Cost represents the cost for a frame to reach the root bridge through the current port. A lower Port-Cost value is preferred over a higher one.

The default Port Cost is 20,000. From cfg-if-gig mode, use the spanning-tree cost command to change the Port Cost for the current port:

spanning-tree cost *cost*

where ***cost*** is a number from 1 to 200,000, where 1 is the lowest cost. The default cost is 20,000.

For example, the following command sequence sets the Port Cost to 100 for the interface at slot 2, port 5:

```
bstnA(cfg)# interface gigabit 2/5
bstnA(cfg-if-gig[2/5])# spanning-tree cost 100
bstnA(cfg-if-gig[2/5])# ...
```

Setting Port Priority

The final step in configuring a port for spanning-tree is to address Port Priority. The Port Priority influences the choice of designated ports in the spanning tree; if multiple ports on the same LAN segment have equal Port Cost (see above), Port Priority is used to break the tie.

Zero (0) is the best possible port priority, 240 is the worst. For cases where more than one port connects to the same LAN segment, set the best priority for the port with the fastest connection. The default is 128.

From cfg-if-gig mode, use the spanning-tree priority command to set the Port Priority for the current port:

spanning-tree priority *priority*

where ***priority*** is a number from 0 to 240, where 0 is the highest priority.

For example, the following command sequence sets the Port Priority to 50 for the interface at port 2/5:

```
bstnA(cfg)# interface gigabit 2/5
bstnA(cfg-if-gig[2/5])# spanning-tree priority 50
bstnA(cfg-if-gig[2/5])# ...
```

Removing a Port from the Spanning-Tree Topology

You can remove a port from participation in the spanning tree. This removes the port from the spanning-tree topology; neighboring bridges will stop using the current port to relay packets to other LAN segments. From cfg-if-gig mode (or cfg-if-ten-gig mode on the ARX-4000), use the spanning-tree shutdown command to remove the current port from spanning-tree service:

spanning-tree shutdown

For example, the following command sequence stops spanning tree on the interface at port 2/4:

```
bstnA(cfg)# interface gigabit 2/4
```

```
bstnA(cfg-if-gig[2/4])# spanning-tree shutdown
bstnA(cfg-if-gig[2/4])# ...
```

Returning to the Spanning-Tree Topology

A port with spanning tree enabled can participate in the spanning tree. It can also be used as the root port for the ARX. By default, all ports have spanning tree enabled. From cfg-if-gig mode, use the no spanning-tree shutdown command to restart spanning-tree on the current port:

```
no spanning-tree shutdown
```

For example, the following command sequence starts spanning tree on the interface at port 2/4:

```
bstnA(cfg)# interface gigabit 2/4
bstnA(cfg-if-gig[2/4])# no spanning-tree shutdown
bstnA(cfg-if-gig[2/4])# ...
```

Configuring RSTP Edge Ports

For installations running RSTP, the final step in spanning-tree configuration is to identify your Edge Ports. Setting a port as an Edge Port indicates that it connects to only one other interface. Edge ports cannot create bridge loops, so RSTP can move an Edge Port from *discarding* state directly to *forwarding* state, skipping the *learning* state. You should set this for all such ports, to increase spanning-tree convergence speed.

By default, ports are not configured as Edge Ports. From cfg-if-gig mode, use the spanning-tree edgeport command to identify the current port as an RSTP Edge Port:

```
spanning-tree edgeport
```

For example, the following command sequence sets the interface at port 2/4 as an Edge Port:

```
bstnA(cfg)# interface gigabit 2/4
bstnA(cfg-if-gig[2/4])# spanning-tree edgeport
bstnA(cfg-if-gig[2/4])# ...
```

Clearing Edge-Port Status

From cfg-if-gig mode, use the no spanning-tree edgeport command to stop treating the current interface as an RSTP Edge Port:

```
no spanning-tree edgeport
```

For example:

```
bstnA(cfg)# interface gigabit 2/6
bstnA(cfg-if-gig[2/6])# no spanning-tree edgeport
bstnA(cfg-if-gig[2/6])# ...
```

Shutting Down Spanning Tree

By default, the ARX runs RSTP when switch-forwarding is enabled. This protects against network loops. You can stop participating in the spanning-tree topology by shutting down the spanning tree. This causes the ARX to relay all BPDU frames instead of processing them, and stops the

ARX from sending BPDUs over its designated ports. The affect on the spanning-tree topology will be the same as if the bridge was removed from the network.

Important

Switch forwarding without spanning tree can possibly result in network loops. Do not shut down spanning tree unless you are sure that your network topology is otherwise insulated from loops.

From cfg-stp mode, use shutdown to shut down the current spanning tree:

```
shutdown
```

For example, the following command sequence shuts down the spanning tree:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# shutdown
bstnA(cfg-stp)# ...
```

Starting Spanning Tree

Spanning tree runs on the ARX by default whenever switch-forwarding is enabled. (If switch forwarding is disabled, the ARX can only participate in a spanning tree as an end station.) If spanning tree has been shut down, you can use the no shutdown command to restart spanning-tree processing:

```
no shutdown
```

For example, the following command sequence starts spanning tree:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# no shutdown
bstnA(cfg-stp)# ...
```

Showing the Spanning-Tree Summary

Use the show spanning-tree summary command to show brief information about the spanning-tree configuration:

```
show spanning-tree summary
```

For example:

```
bstnA(cfg)# show spanning-tree summary

Spanning Tree Admin State is Disabled
Configuration: Protocol          : IEEE_Dot1d
               Revision Level    : 0
               Format Selector   : 0
Default      : Name              : 00-0A-49-17-78-40
Total MST Instances Created      : 0
bstnA(cfg)# ...
```

Showing Spanning-Tree Details

Use the show spanning-tree detailed command for details of the spanning-tree configuration:

```
show spanning-tree detailed
```

For example:

```
bstnA(cfg)# show spanning-tree detailed

Bridge is Executing the IEEE compatible IEEE_Dot1d Spanning Tree protocol

Switch STP Admin State     Disabled
Bridge Priority            61440
Bridge Address             00:0a:49:17:78:40
Bridge Max Age             20 sec
Bridge Hello Time          2 sec
Bridge Forward Delay       15 sec
Bridge Hold Time           3 sec

Designated Root: Priority 240
                 Address  00:0a:49:17:78:40
Root Path Cost             0
Root Port Max Age          20
Root Port Fwd Delay        15
Time Since Topology Change 6938 sec
Topology Change Count      0
Topology Change            0


Slot/     Spanning Tree                  Port     BPDU Packets
Port    Forwarding    Admin    Path Cost Type    Rx          Tx
---- --------------- -------- ---------- ---- ---------- -----------
2/5  Manual Forward  Disabled 20000      gbe  0          0
2/6  Manual Forward  Disabled 20000      gbe  0          0

bstnA(cfg)# ...
```

Showing Spanning-Tree Port Configuration

You can display the port-level spanning-tree parameters for an interface, such as Port Priority and Port Cost. Use the show spanning-tree interface command:

show spanning-tree interface *slot/port*

where ***slot/port*** defines a physical Ethernet port (for example, “2/7” on an ARX-4000 or “1/2” on an ARX-2000).

For example, the following command shows the spanning-tree parameters for the interface at port 2/3:

```
bstnA(cfg)# show spanning-tree interface 2/3

Bridge is Executing the IEEE compatible Spanning Tree protocol

Slot                    2
Interface               3
Port: SNMP ID           32785
      STP  ID           17
      Priority          128
      Forwarding State  Disabled
      STP State         Disabled
      Role              N/A
      Path cost         20000
Designated: SNMP Port ID 0
            STP Port ID  0
            Priority     0
            Address      00:0a:49:17:78:40
Edge Port: Admin Status      Configured
```

```
           Operational Status Operational
Point to Point Mac Status     Auto
Topology Change Ack     0
Port Up Time            6939 sec
STP BPDU Transmitted    0
STP BPDU Received       0
RST BPDU Transmitted    0
RST BPDU Received       0
MST BPDU Transmitted    0
MST BPDU Received       0
```

Deactivating Switch Forwarding (and STP)

To stop the ARX from forwarding packets between client/server ports, use the no switch-forwarding enable command from cfg mode:

```
no switch-forwarding enable
```

This also shuts down spanning tree. The CLI prompts for confirmation; enter **yes** to proceed.

For example:

```
bstnA(cfg)# no switch-forwarding enable
Warning: Disable switch forwarding between all Ethernet Interfaces?
This will also disable the Spanning Tree Protocol. [yes/no] yes
bstnA(cfg)# ...
```

Configuring a Layer-2 Interface

A layer-2 port is called an *interface* in the CLI. This section describe gigabit interfaces, the most-common interface in all of the hardware-based ARX platforms. Each gigabit interface starts with the following defaults:

- speed = auto
- flow control disabled
- shut down

You can change these defaults with the commands in cfg-if-gig mode. From cfg mode, use the interface gigabit command to select an interface for layer-2 configuration:

```
interface gigabit slot/port
```

where ***slot/port*** defines a physical Ethernet port (for example, "2/6"). Use the show interface summary command to find all ports on the switch. Choose a gigabit interface, labeled "gbe" in the output.

This places you in cfg-if-gig mode, where you can set various configuration parameters or enable/disable the interface.

For example, the following command sequence enters cfg-if-gig mode for the gigabit interface at port 2/3 (the first single-gigabit interface on an ARX-4000).

```
bstnA(cfg)# interface gigabit 2/3
bstnA(cfg-if-gig[2/3])# ...
```

Setting Speed

By default, a gigabit interface negotiates its speed with its neighbor(s) when it starts, choosing the highest speed possible for both ends of the connection. It also discovers the line type (fast Ethernet or fiber Ethernet) and duplex configuration (half or full) automatically. For situations where automatic negotiation is unreliable, you have the option to set the speed, line type, and duplex configuration manually.

The speed cannot be set to auto for the port to join a channel (channels are explained later in the chapter).

From cfg-if-gig mode, use the speed command to set the interface speed, line type, and duplex configuration.

```
speed {auto | 100-tx-half | 100-tx-full | 100-fx-full | 1000-full}
```

where **{auto | 100-tx-half | 100-tx-full | 100-fx-full | 1000-full}** is a required choice:

auto is auto-negotiate, the default.

100-tx-half is fast Ethernet, 100 megabits-per-second (mbps, where a megabit is 1,000,000 bits), half duplex.

100-tx-full is fast Ethernet, 100 mbps, half duplex.

100-fx-full is fiber Ethernet, 100 mbps, full duplex.

1000-full is fiber or copper Ethernet, 1000 mbps, full duplex.

For the interface at port 2/5, the following command sequence sets the interface speed to 100 mbps, the line type to fiber Ethernet, and the duplex configuration to full duplex.

```
bstnA(cfg)# interface gigabit 2/5
bstnA(cfg-if-gig[2/5])# speed 100-fx-full
bstnA(cfg-if-gig[2/5])# ...
```

Enabling Flow Control

The next step in configuring an interface is addressing flow control. If an interface with flow control is overloaded, it can send a flow-control request to its peer(s). If a peer is overloaded, it can send a flow-control request to the local interface. The flowcontrol command determines whether or not the current interface sends or receives flow-control requests. By default, flow control is disabled for all interfaces; you can only enable flow control for an interface where the speed is set to 1000-full.

From cfg-if-gig mode, use the flowcontrol command to configure flow control:

```
flowcontrol send {off | on}
flowcontrol receive {off | on}
```

where **{off | on}** is a required choice in both cases.

For example, the following command sequence enables symmetric flow control (both send and receive) for the interface at port 2/3:

```
bstnA(cfg)# interface gigabit 2/3
bstnA(cfg-if-gig[2/3])# flowcontrol send on
bstnA(cfg-if-gig[2/3])# flowcontrol receive on
bstnA(cfg-if-gig[2/3])# ...
```

Disabling Flow Control

Flow control is disabled by default for both send and receive. From cfg-if-gig mode, use the no flowcontrol command to disable flow control:

```
no flowcontrol
```

For example, the following command sequence disables flow control for the interface at port 2/3:

```
bstnA(cfg)# interface gigabit 2/3
bstnA(cfg-if-gig[2/3])# no flowcontrol
bstnA(cfg-if-gig[2/3])# ...
```

Traffic-Storm Control

A traffic storm is a flood of frames in a short period of time. To prevent any port from forwarding the traffic storm, the ARX forwards a maximum of 1000 packets per second for each of the following frame types: broadcast,

multicast, or unicast frames with unknown destination addresses. If the ingress-side of the port reaches the maximum in a given second, the port drops packets of the chosen type until the second is over.

Traffic-storm control does not suppress spanning-tree BPDU packets.

This parameter is not configurable.

Setting an Interface Description (optional)

An interface description is an optional label that appears in the show interface gigabit output. From cfg-if-gig mode, use the description command to add a description:

description *interface-description*

where ***interface-description*** can have up to 60 characters. Quote the description if it contains spaces.

For example, the following command sequence describes the interface at port 2/3:

```
bstnA(cfg)# interface gigabit 2/3
bstnA(cfg-if-gig[2/3])# description beLink
bstnA(cfg-if-gig[2/3])# ...
```

Removing the Interface Description

From cfg-if-gig mode, use no description to remove the description from the current interface configuration:

no description

For example:

```
bstnA(cfg)# interface gigabit 2/4
bstnA(cfg-if-gig[2/4])# no description
bstnA(cfg-if-gig[2/4])# ...
```

Starting an Interface

All interfaces are shut down by default. From cfg-if-gig mode, use the no shutdown command to start the current interface:

no shutdown

For example, the following command sequence starts the interface at port 2/3:

```
bstnA(cfg)# interface gigabit 2/3
bstnA(cfg-if-gig[2/3])# no shutdown
bstnA(cfg-if-gig[2/3])# ...
```

Shutting Down an Interface

You can stop traffic on an interface by shutting it down. From cfg-if-gig mode, use shutdown to shut down the current interface:

shutdown

For example, the following command sequence shuts down the interface at port 2/9:

```
bstnA(cfg)# interface gigabit 2/9
bstnA(cfg-if-gig[2/9])# shutdown
bstnA(cfg-if-gig[2/9])# ...
```

Configuring a Ten-Gigabit Interface (ARX-2500 and ARX-4000 Only)

The ARX-2500 and ARX-4000 each have two ten-gigabit interfaces. Skip this section unless you have an ARX-2500 or ARX-4000.

Each ten-gigabit interface starts with the following settings and defaults:

- speed = ten-gigabit, full-duplex (non-configurable)
- flow control disabled
- shut down

You can change these defaults with the commands in cfg-if-ten-gig mode. From cfg mode, use the interface ten-gigabit command to select a ten-gigabit interface for layer-2 configuration:

interface ten-gigabit *slot/port*

where ***slot/port*** defines a ten-gigabit Ethernet port (2/1 or 2/2). Use the show interface summary command to find all ports on the switch. Choose a ten-gigabit interface, labeled “10gbe” in the output.

This places you in cfg-if-ten-gig mode, where you can set various configuration parameters or enable/disable the interface.

For example, the following command sequence enters cfg-if-ten-gig mode for the ten-gigabit interface at slot 2, port 1.

```
bstnA(cfg)# interface ten-gigabit 2/1
bstnA(cfg-if-ten-gig[2/1])# ...
```

Enabling Flow Control

The next step in configuring a ten-gigabit interface is addressing flow control. As described above for a gigabit interface, a ten-gigabit interface can send a flow-control request to its peer(s) if it is overloaded. Conversely, the peer can send a flow-control request to the local interface. The flowcontrol command determines whether or not the current interface sends or receives flow-control requests. By default, flow control is disabled for all interfaces.

From cfg-if-ten-gig mode, use the flowcontrol command to configure flow control:

flowcontrol send {off | on}

flowcontrol receive {off | on}

where **{off | on}** is a required choice in both cases.

For example, the following command sequence enables symmetric flow control (both send and receive) for the interface at slot 2, port 1:

```
bstnA(cfg)# interface ten-gigabit 2/1
bstnA(cfg-if-ten-gig[2/1])# flowcontrol send on
bstnA(cfg-if-ten-gig[2/1])# flowcontrol receive on
bstnA(cfg-if-ten-gig[2/1])# ...
```

Disabling Flow Control

Flow control is disabled by default for both send and receive. From cfg-if-ten-gig mode, use the no flowcontrol command to disable flow control:

```
no flowcontrol
```

For example, the following command sequence disables flow control for the interface at slot 2, port 1:

```
bstnA(cfg)# interface ten-gigabit 2/1
bstnA(cfg-if-ten-gig[2/1])# no flowcontrol
bstnA(cfg-if-ten-gig[2/1])# ...
```

Traffic-Storm Control

A traffic storm is a flood of frames in a short period of time. To prevent any port from forwarding the traffic storm, the ARX forwards a maximum of 1000 packets per second for each of the following frame types: broadcast, multicast, or unicast frames with unknown destination addresses. If the ingress-side of the port reaches the maximum in a given second, the port drops packets of the chosen type until the second is over.

Traffic-storm control does not suppress spanning-tree BPDU packets.

This parameter is not configurable.

Setting an Interface Description (optional)

An interface description is an optional label that appears in the show interface ten-gigabit output, described later. From cfg-if-ten-gig mode, use the description command to add a description:

```
description interface-description
```

where ***interface-description*** can have up to 60 characters. Quote the description if it contains spaces.

For example, the following command sequence describes the interface at slot 2, port 1:

```
bstnA(cfg)# interface ten-gigabit 2/1
bstnA(cfg-if-ten-gig[2/1])# description "WAN uplink"
bstnA(cfg-if-ten-gig[2/1])# ...
```

Removing the Interface Description

From cfg-if-ten-gig mode, use no description to remove the description from the current interface configuration:

```
no description
```

For example:

```
bstnA(cfg)# interface ten-gigabit 2/2
bstnA(cfg-if-ten-gig[2/2])# no description
bstnA(cfg-if-ten-gig[2/2])# ...
```

Starting a Ten-Gigabit Interface

All interfaces are shut down by default. From cfg-if-ten-gig mode, use the no shutdown command to start the current ten-gigabit interface:

```
no shutdown
```

For example, the following command sequence starts the interface at slot 2, port 1:

```
bstnA(cfg)# interface ten-gigabit 2/1
bstnA(cfg-if-ten-gig[2/1])# no shutdown
bstnA(cfg-if-ten-gig[2/1])# ...
```

Shutting Down a Ten-Gigabit Interface

You can stop traffic on an interface by shutting it down. From cfg-if-ten-gig mode, use shutdown to shut down the current ten-gigabit interface:

```
shutdown
```

For example, the following command sequence shuts down the interface at slot 2, port 2:

```
bstnA(cfg)# interface ten-gigabit 2/2
bstnA(cfg-if-ten-gig[2/2])# shutdown
bstnA(cfg-if-ten-gig[2/2])# ...
```

Showing Interface Configuration

Use the show interface gigabit command to see the configuration for a particular interface:

```
show interface gigabit slot/port
```

where ***slot/port*** identifies a physical Ethernet port (for example, “2/8”). Use the show interface summary command to find all ports on the switch.

For example, the following command shows the interface at slot 2, port 3:

```
bstnA(cfg)# show interface gigabit 2/3

Slot                    2
Interface               3
Description             Default
Type                    Copper
Mode                    Normal
Admin State             Enabled
Link Status             Down
Speed                   1 Gb/s
Duplex                  Unknown
Auto Negotiation(Admin) Enabled
Flow Control(Admin)
             Receive    Off
             Send       Off
MAC Address             00:0a:49:17:78:34
```

```
Storm Control:Broadcast  1000 packets/sec
              Multicast  1000 packets/sec
              Unknown DA 1000 packets/sec
Port VLAN ID             0
Accept Frames            Admit All
bstnA(cfg)# ...
```

Showing a Ten-Gigabit Interface (ARX-2500 and ARX-4000 Only)

On an ARX-2500 or ARX-4000, you can show the configuration of a ten-gigabit interface. Use the show interface ten-gigabit command to see the configuration parameters:

show interface ten-gigabit *slot/port*

where ***slot/port*** (2/1 or 2/2) identifies a physical Ethernet port. Use the show interface summary command to find all ports on the switch.

For example, the following command shows the ten-gigabit interface at slot 2, port 2:

```
bstnA(cfg)# show interface ten-gigabit 2/2

Slot                     2
Interface                2
Description              Default
Type                     10GBASE-SR X2
Mode                     Normal
Admin State              Enabled
Link Status              Down
Speed                    10 Gb/s
Duplex                   Unknown
Auto Negotiation(Admin)  Disabled
Flow Control(Admin)
             Receive     Off
             Send        Off
MAC Address              00:0a:49:17:78:33

Storm Control:Broadcast  1000 packets/sec
              Multicast  1000 packets/sec
              Unknown DA 1000 packets/sec
Port VLAN ID             0
Accept Frames            Admit All
bstnA(cfg)# ...
```

Showing Interface Statistics

Add an optional stats keyword to see counters for the various frames received (ingress) and transmitted (egress) over a particular interface:

show interface gigabit *slot/port* stats

where ***slot/port*** identifies a physical Ethernet port (for example, “2/6”). Use the show interface summary command to find all ports on the switch.

For example, the following command shows the interface statistics at slot 2, port 5:

```
bstnA(cfg)# show interface gigabit 2/5 stats

Slot                          2
```

```
Interface Id                  5
------------------------------------------------------------------
                               Ingress                  Egress
------------------------------------------------------------------

Octets                       731660387                633394831
Total Frames                   1061953                  1134861
Unicast Frames                 1047897                  1134271
Multicast Frames                  3048                        0
Broadcast Frames                 11008                      590
PAUSE Frames                         0                        0
CRC Errors                           0
Total Discards                       0
Alignment Errors                     0
Single Collisions                                             0
Multiple Collisions                                           0
Late Collisions                                               0
Excessive Collisions                                          0

------------------------------------------------------------------
                               Ether Stats
------------------------------------------------------------------

Packet Size:64                 126362
            65-127              21677
            128-255           1221543
            256-511             40980
            512-1023           101740
            1024-1518          684512
            > 1519                  0
Total Collisions                    0
bstnA(cfg)# ...
```

Showing Ten-Gigabit-Interface Statistics

You can also add the optional stats keyword to the end of the show interface ten-gigabit command:

show interface ten-gigabit *slot/port* stats

where ***slot/port*** (2/1 or 2/2) identifies a ten-gigabit port. Use the show interface summary command to find all ports on the switch; the ten-gigabit ports are labeled “10gbe.”

The output has the same statistics as the show interface gigabit stats output.

For example, the following command shows the statistics at slot 2, port 2:

```
bstnA(cfg)# show interface ten-gigabit 2/2 stats

Slot                          2
Interface Id                  2
------------------------------------------------------------------
                               Ingress                  Egress
------------------------------------------------------------------

Octets                               0                        0
Total Frames                         0                        0
Unicast Frames                       0                        0
Multicast Frames                     0                        0
Broadcast Frames                     0                        0
```

```
PAUSE Frames                          0                       0
CRC Errors                            0
Total Discards                        0
Alignment Errors                      0
Single Collisions                                             0
Multiple Collisions                                           0
Late Collisions                                               0
Excessive Collisions                                          0

----------------------------------------------------------------------
                                Ether Stats
----------------------------------------------------------------------

Packet Size:64                        0
            65-127                    0
            128-255                   0
            256-511                   0
            512-1023                  0
            1024-1518                 0
            > 1519                    0
Total Collisions                      0
bstnA(cfg)# ...
```

Clearing Interface Statistics

From priv-exec mode, you can use the clear counters gigabit and/or clear counters ten-gigabit command to bring all counters back to zero on all external interfaces:

clear counters gigabit

clear counters ten-gigabit

To clear only one interface, specify its slot and port at the end of the command:

clear counters gigabit [*slot/port*]

clear counters ten-gigabit [*slot/port*]

where ***slot/port*** (optional) identifies one physical Ethernet port (for example, “3/6,” “2/2,” or “1/1”). Use the show interface summary command to find all ports on the switch.

For example, the following command exits to priv-exec mode, clears the counters for interface 1/5, shows the interface statistics at zero, then shows the statistics again to watch them grow:

```
prtlndA(cfg)# exit
prtlndA# clear counters gigabit 1/5
prtlndA# show interface gigabit 1/5 stats


Slot                          1
Interface Id                  5
----------------------------------------------------------------------
                                Ingress                 Egress
----------------------------------------------------------------------

Octets                                0                       0
Total Frames                          0                       0
Unicast Frames                        0                       0
Multicast Frames                      0                       0
Broadcast Frames                      0                       0
```

```
PAUSE Frames                          0                       0
CRC Errors                            0
Total Discards                        0
Alignment Errors                      0
Single Collisions                                             0
Multiple Collisions                                           0
Late Collisions                                               0
Excessive Collisions                                          0

-------------------------------------------------------------------
                                Ether Stats
-------------------------------------------------------------------

Packet Size:64                        0
            65-127                    0
            128-255                   0
            256-511                   0
            512-1023                  0
            1024-1518                 0
            > 1519                    0
Total Collisions                      0
prtlndA# ...
prtlndA# show interface gigabit 1/5 stats

Slot                          1
Interface Id                  5
-------------------------------------------------------------------
                               Ingress                  Egress
-------------------------------------------------------------------

Octets                       595866453                249378595
Total Frames                    462371                   299171
Unicast Frames                  438085                   298916
Multicast Frames                  4578                        0
Broadcast Frames                 19708                      255
PAUSE Frames                         0                        0
CRC Errors                           0
Total Discards                       0
Alignment Errors                     0
Single Collisions                                             0
Multiple Collisions                                           0
Late Collisions                                               0
Excessive Collisions                                          0

-------------------------------------------------------------------
                                Ether Stats
-------------------------------------------------------------------

Packet Size:64                   28659
            65-127              111869
            128-255              56875
            256-511              14163
            512-1023              8470
            1024-1518           541506
            > 1519                   0
Total Collisions                     0
prtlndA# ...
```

Setting the MAC-Address Aging Time

The ARX learns its MAC addresses by examining the source MAC addresses of incoming frames. Previously-unknown MACs are added to (or updated in) an internal Forwarding DataBase (*FDB*). The ARX keeps a limited number of MAC addresses in the FDB, and removes any address that exceeds the *Aging Time* that you set for the switch. This learning process, the FDB, and the Aging Time are all defined in IEEE 802.1D.

The default Aging Time is 300 (seconds, or five minutes). From cfg-stp mode, use the mac-address aging-time command to reset the Aging Time for the ARX:

```
mac-address aging-time seconds
```

where ***seconds*** (300-1,000,000) is the new Aging Time.

For example, the following command sequence sets the Aging Time to 600 seconds (10 minutes):

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# mac-address aging-time 600
bstnA(cfg-stp)# ...
```

Reverting to the Default Aging Time

The default MAC-address Aging Time is 300 (seconds, or five minutes). From cfg-stp mode, use the no mac-address aging-time command to revert to this default:

```
no mac-address aging-time
```

For example:

```
bstnA(cfg)# spanning-tree
bstnA(cfg-stp)# no mac-address aging-time
bstnA(cfg-stp)# ...
```

Showing the MAC-Address Table

Use the show mac-address-table command to view the switch's table of MAC addresses:

```
show mac-address-table
```

For each known MAC address, this shows the slot and port where the address was learned (if any), the VLAN associated with the MAC, the Channel ID for the MAC (if any), and the Mode in which the MAC was learned. The Mode can be Learned (from external traffic), Management (associated with the MGMT interface on the ARX front panel, or the management address advertised in the spanning-tree protocol), Inband (associated with an inband-management interface, described in a later chapter), or Self (internally-assigned).

For example:

```
bstnA> show mac-address-table
```

```
Slot Port  MAC Address         VLAN ID    Channel ID    Mode
-----------------------------------------------------------------
         00:0a:49:17:78:ff        25                    Inband
         00:0a:49:17:78:fe        25                    Inband
         00:0a:49:17:78:40         1                    Management
   2   5 00:01:e8:5e:ea:1f        25                    Learned
   2   5 00:0a:49:17:74:c0        25                    Learned
   2   5 00:0a:49:17:74:ff        25                    Learned
   2   5 00:0a:49:17:79:ff        25                    Learned
   2   5 00:0a:49:17:80:ff        25                    Learned
   2   5 00:0a:49:17:81:c0        25                    Learned
   2   5 00:0a:49:17:81:ff        25                    Learned
   2   5 00:0a:49:17:86:c0        25                    Learned
   2   5 00:0a:49:17:86:ff        25                    Learned
   2   5 00:0a:49:17:8c:c0        25                    Learned
...
```

Showing a Summary of the MAC-Address Table

Use the show mac-address-table summary command for a summary view of the MAC-address table:

show mac-address-table summary

For example:

```
bstnA> show mac-address-table summary

Mac Address Learning Mode      IVL (Independent Vlan Learning)
MAC Address High Count           163
Active MAC Addresses in FDB      163
Maximum MAC Addresses Supported  12000
Configured Aging Time            300 secs
...
```

Configuring a Channel (802.3ad Link Aggregation)

The ARX-500 has a single client/server port and does not support the use of channels. Skip this section if you are setting up an ARX-500 chassis.

Link aggregation, specified in IEEE 802.3ad, combines multiple Ethernet ports into a single aggregated flow. Each aggregate is a *channel*, identified by an integer ID. The station on the other side of the channel must have the same members and VLAN(s) as the ARX; traffic will not flow through the member ports unless the channel configurations match.

If spanning tree is running, the channel's VLAN determines its spanning-tree membership.

You can configure up to eight channels on an ARX. From cfg mode, use the channel command to create a new channel configuration:

```
channel id
```

where ***id*** (1-8) is an integer that you choose for the channel.

This places you in cfg-channel mode, where you must choose member ports for the channel and then activate it. You can also set some optional parameters in cfg-channel mode, such as a description string.

For example, the following command sequence creates channel 1:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# ...
```

Channels are active upon creation; there is no need to execute the no shutdown command.

For ARX-1500 and ARX-2500, a channel is not added to any VLAN by default. The VLAN for each channel must be configured explicitly.

Note

When redundant ARX pairs are in use, F5 Networks recommends strongly that you configure a link-aggregation channel for the redundancy link to ensure resilient and optimized performance.

Adding the Channel to a VLAN

By default, a new channel is assigned to VLAN 1, the default VLAN for traffic to clients and servers.

Note

For ARX-1500 and ARX-2500, a channel is not assigned to a VLAN by default; you must configure the VLAN explicitly.

For ARX-1500 and ARX-2500, a non-default VLAN can include only one member interface.

From cfg-channel mode, use the vlan command to assign the current channel to another VLAN:

vlan *vlan-id*

where ***vlan-id*** (1-4095) identifies the new VLAN for this channel.

For example, the following command sequence assigns channel 3 to VLAN 6:

```
bstnA(cfg)# channel 3
bstnA(cfg-channel[3])# vlan 6
bstnA(cfg-channel[3])# ...
```

Setting the Channel for VLAN Tagging

The vlan command adds the channel to a VLAN with tagging disabled; that is, outgoing frames are not tagged with a VLAN ID, and the channel can therefore support only one VLAN. To enable VLAN tagging for the current channel (thereby allowing the channel to support multiple VLANs), use the vlan-tag command:

vlan-tag *vlan-id*

where ***vlan-id*** identifies the new VLAN for this channel.

For example, the following command sequence assigns channel 3 to VLANs 6 and 7, with tagging enabled:

```
bstnA(cfg)# channel 3
bstnA(cfg-channel[3])# vlan-tag 6
bstnA(cfg-channel[3])# vlan-tag 7
bstnA(cfg-channel[3])# ...
```

Removing the Channel from a VLAN

From cfg-channel mode, use the no vlan-tag command to remove the current channel from a tagged VLAN:

no vlan-tag *vlan-id*

where ***vlan-id*** identifies the VLAN to remove.

If you remove the last VLAN from a channel, the channel reverts to VLAN 1.

For example, the following command sequence removes channel 3 from VLAN 7:

```
bstnA(cfg)# channel 3
bstnA(cfg-channel[3])# no vlan-tag 7
bstnA(cfg-channel[3])# ...
```

Reverting a No-Tagging Channel to the Default VLAN

From cfg-channel mode, use the no vlan command to send the current channel back to its default VLAN, VLAN 1:

no vlan *vlan-id*

where ***vlan-id*** identifies the VLAN to remove.

This reverts the channel to VLAN 1.

For example, the following command sequence removes channel 3 from VLAN 7:

```
bstnA(cfg)# channel 3
bstnA(cfg-channel[3])# no vlan 7
bstnA(cfg-channel[3])# ...
```

Preparing to Add Member Ports

Before you add any member ports to the channel, we recommend that you disconnect them from the network. Once the channel is configured on the ARX *and* the switch at the other end of the channel, you can connect the switches.

Preparing a Channel as a Redundant-Pair Link

For a channel to be used as the redundant-pair link between two peers, the next step is to prepare the channel for redundancy and add member ports to it. If this channel is not intended as the redundant-pair link, continue to the next section.

◆ Note

When redundant ARX pairs are in use, F5 Networks recommends strongly that you configure a link-aggregation channel for the redundancy link to ensure resilient and optimized performance.

◆ Note

These instructions do not apply to ARX-1500 and ARX-2500, for which the redundant-pair link is a VLAN. In this case, a channel can be associated with a VLAN, and that VLAN prepared in turn for use in a redundant-pair link, as described in Preparing for Use in a Redundant-Pair Link, on page 3-53.

A channel can have up to 8 ports. All member ports must be in the same VLAN as the channel, they must be running full-duplex, and they must run at the same speed. A port with auto-detect settings *cannot* be a member of the channel. Refer back to *Setting Speed*, on page 3-23 to set the port speeds.

From cfg-channel mode, use the redundancy protocol command to add member ports to the channel and enable it for use as a redundant-pair link:

redundancy protocol *slot/port*

where ***slot/port*** defines a physical port on an NSM (for example, “3/3”),

or

redundancy protocol *slot/port* to *slot/port*

where ***slot/port* to *slot/port*** defines a range of physical Ethernet ports. For example, “2/5 to 2/12” is valid on an ARX-4000, and “1/2 to 1/4” is valid on an ARX-2000. If you want to use the ten-gigabit ports on the ARX-4000, use “2/1 to 2/2.” Use the show interface summary command to find all ports on the ARX.

You can use the redundancy protocol command multiple times to define more than one range of ports.

◆ Note

If any of the channel's ports is an active member of a spanning tree, the channel assumes the same responsibility (as a root or designated port) in the spanning tree. It uses all of that port's spanning-tree parameters, such as Port Cost and Port Priority.

For example, the following command sequence takes place on two ARX-2000 peers with a channel connection. For Peer A, "prtlndA," the command sequence creates a channel from two gigabit ports and tags the channel properly. LACP is enabled on the channel as well. The process is then repeated (using different ports) at Peer B, "prtlndB:"

Peer A (ports 1/5 and 1/6):

```
prtlndA(cfg)# interface gigabit 1/5
prtlndA(cfg-if-gig[1/5])# speed 1000-full
prtlndA(cfg-if-gig[1/5])# no shutdown
prtlndA(cfg-if-gig[1/5])# exit
prtlndA(cfg)# interface gigabit 1/6
prtlndA(cfg-if-gig[1/6])# speed 1000-full
prtlndA(cfg-if-gig[1/6])# no shutdown
prtlndA(cfg-if-gig[1/6])# exit
prtlndA(cfg)# channel 1
prtlndA(cfg-channel[1])# redundancy protocol 1/5 to 1/6
prtlndA(cfg-channel[1])# lacp passive
prtlndA(cfg-channel[1])# exit
prtlndA(cfg)# ...
```

Peer B (ports 1/2 and 1/3):

```
prtlndB(cfg)# interface gigabit 1/2
prtlndB(cfg-if-gig[1/2])# speed 1000-full
prtlndB(cfg-if-gig[1/2])# no shutdown
prtlndB(cfg-if-gig[1/2])# exit
prtlndB(cfg)# interface gigabit 1/3
prtlndB(cfg-if-gig[1/3])# speed 1000-full
prtlndB(cfg-if-gig[1/3])# no shutdown
prtlndB(cfg-if-gig[1/3])# exit
prtlndB(cfg)# channel 1
prtlndB(cfg-channel[1])# redundancy protocol 1/2 to 1/3
prtlndB(cfg-channel[1])# lacp passive
prtlndB(cfg-channel[1])# exit
prtlndB(cfg)# ...
```

This prepares the channels for use in a redundant-pair connection. The redundancy-configuration chapter, below, describes how to start redundancy processing.

Removing a Port from its Redundant-Pair-Link Channel

From cfg-channel mode, use the no redundancy protocol command to remove one or more ports from the current redundancy-ready channel:

```
no redundancy protocol slot/port
```

where ***slot/port*** **to** ***slot/port*** defines a port to remove (for example, "2/6"), or

no redundancy protocol *slot/port* to *slot/port*

where ***slot/port* to *slot/port*** defines a range of ports to remove (for example, “2/3 to 2/6”).

To bring the channel down to a single member, you must remove the channel configuration altogether. This process is described later in this section.

You can use the no redundancy protocol command multiple times to remove more than one port or multiple port ranges.

For example, the following command sequence removes two ports from channel 3:

```
bstnA(cfg)# channel 3
bstnA(cfg-channel[3])# no redundancy protocol 2/9 to 2/10
bstnA(cfg-channel[3])# ...
```

Adding Ports to a Standard Channel

The next step in configuring a standard channel (that is, a channel that cannot be used in a redundant-pair link) is to identify the ports that belong to it. If this is a redundant-pair-link channel and you have already added its ports, you can skip this section.

The same rules for member ports apply to a standard channel: the channel can have up to 8 ports, all the ports must be in the channel’s VLAN, and all ports must be running full-duplex at the same speed. A port with auto-detect settings *cannot* be a member of the channel; see *Setting Speed*, on page 3-23 to set the port speeds.

From cfg-channel mode, use the members command to define a range of physical ports that belong to the current channel:

members *slot/port*

where ***slot/port*** defines a physical port (for example, “2/3”),

or

members *slot/port* to *slot/port*

where ***slot/port* to *slot/port*** defines a range of physical Ethernet ports. For example, “2/3 to 2/11” is a valid range on an ARX-4000, and “1/2 to 1/4” is valid on an ARX-2000. Use the show interface summary command to find all ports on the ARX.

The ARX-2500 and ARX-4000 have two port types: ten-gigabit and gigabit. You can combine both port types in the same channel. Confirm that the peer device (at the other end of the channel) supports mixed speeds before you configure this.

You can use the members command multiple times to define more than one range of ports.

◆ Note

If any of the channel's ports is an active member of a spanning tree, the channel assumes the same responsibility (as a root or designated port) in the spanning tree. It uses all of that port's spanning-tree parameters, such as Port Cost and Port Priority.

For example, the following command sequence prepares two ports with matching speeds and then aggregates them into channel 3:

```
prtlndA(cfg)# interface gigabit 1/3
prtlndA(cfg-if-gig[1/3])# speed 1000-full
prtlndA(cfg-if-gig[1/3])# no shutdown
prtlndA(cfg-if-gig[1/3])# exit
prtlndA(cfg)# interface gigabit 1/4
prtlndA(cfg-if-gig[1/4])# speed 1000-full
prtlndA(cfg-if-gig[1/4])# no shutdown
prtlndA(cfg-if-gig[1/4])# exit
prtlndA(cfg)# channel 3
prtlndA(cfg-channel[3])# members 1/3 to 1/4
prtlndA(cfg-channel[3])# ...
```

As another example, this next command sequence sets two port ranges for channel 1. (Presume all ports have already had their speeds reset.)

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# members 2/5 to 2/6
bstnA(cfg-channel[1])# members 2/13 to 2/14
bstnA(cfg-channel[1])# ...
```

Removing a Port From its Channel

From cfg-channel mode, use the no members command to remove one or more ports from the current channel:

no members *slot/port*

where ***slot/port* to *slot/port*** defines a port to remove (for example, "2/14"), or

no members *slot/port* to *slot/port*

where ***slot/port* to *slot/port*** defines a range of ports to remove (for example, "2/3 to 2/6")."

To bring the channel down to a single member, you must remove the channel configuration altogether. This process is described later in this section.

You can use the no members command multiple times to remove more than one port or multiple port ranges.

For example, the following command sequence removes four members from channel 1:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# no members 2/5 to 2/6
bstnA(cfg-channel[1])# no members 2/9 to 2/10
bstnA(cfg-channel[1])# ...
```

Enabling LACP

The next step in aggregating links is to address the Link Aggregation Control Protocol (LACP). LACP, defined in IEEE 802.3ad, is a control protocol for managing the channel. By default, this protocol is disabled in a new channel; member traffic is aggregated but LACP packets are ignored. The ARX and the peer device can use LACP to dynamically manage the channel's bandwidth and redundancy, and automatically remove member ports if anyone mistakenly makes a configuration change that disqualifies them for membership.

The peers accomplish this by grouping their member ports, and possibly designating one or more of them as *standby* (currently unused). Without LACP, a disqualifying configuration change may go undetected and cause network issues that are difficult to diagnose.

If LACP is disabled, all member ports remain in the channel no matter what configuration or topology changes occur later.

Enabling LACP On ARX-500, ARX-2000, and ARX-4000

Enabling LACP on older ARX platforms requires you to use the lacp passive command to accept LACP packets at the ARX end of the channel, and to send proper LACP responses. We recommend this if the peer device supports LACP:

```
lacp passive
```

For example, the following command sequence enables LACP for channel 1.

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# lacp passive
bstnA(cfg-channel[1])# ...
```

To establish LACP on the channel, use lacp active at the peer and use lacp passive on the ARX. If you connect two ARX peers over a channel in a redundant configuration, use lacp passive on both ARX peers to establish LACP; one of them assumes the active LACP role automatically.

Enabling LACP On ARX-1500 and ARX-2500

Enabling LACP on ARX-1500 and ARX-2500 requires you to use the lacp active command to accept LACP packets at the ARX end of the channel, and to send proper LACP responses. We recommend this if the peer device supports LACP:

```
lacp active
```

For example, the following command sequence enables LACP for channel 1.

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# lacp active
bstnA(cfg-channel[1])# ...
```

To establish LACP on the channel, use lacp passive at the peer and use lacp active on the ARX. If you connect two ARX peers over a channel in a redundant configuration, use lacp active on both ARX peers to establish LACP; one of them assumes the passive LACP role automatically.

Changing the LACP Timeout

By default, LACP has a 1-second timeout for receiving link status from a partner. The configuration mode CLI command lacp rate long-timeout can be used to change the timeout to 30 seconds. Executing the negative version of the command, no lacp rate, returns the timeout value to its 1-second default.

The command syntax is:

```
lacp rate long-timeout
```

For example:

```
lacp rate long-timeout
```

Setting the System Priority (optional)

The stations at both ends of an LACP channel each have a System Priority, defined in IEEE 802.3ad. Lower numbers indicate high priority; for instance, 1 is an extremely high System Priority. The device with the better (lower-numbered) priority can change the roles of member ports from active to standby, based on port configuration and the channel's usage in the layer-2 network.

This priority can be different for each channel.

The default is 32,767. From cfg-channel mode, use the priority command to set the System Priority for the current channel:

```
priority number
```

where ***number*** (0-65536) is the System Priority that you choose.

All member ports inherit this system priority as their port priorities.

For example, the following command sequence sets the System Priority to 100 on channel 1:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# priority 100
bstnA(cfg-channel[1])# ...
```

Reverting to the Default System Priority

From cfg-channel mode, use no priority to revert back to the default priority, 32,767, for channel 1 on "bstnA:"

```
no priority
```

For example:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# no priority
bstnA(cfg-channel[1])# ...
```

Disabling LACP

You can shut down LACP to stop responding to LACP packets from the peer switch. This stops all dynamic reconfiguration of the channel, though it allows basic aggregation to continue.

Important

This causes all member ports to restart, and therefore results in a brief traffic outage. If possible, avoid turning off LACP when client traffic is flowing through the channel.

Important

If this channel is used as a redundant-pair link (recall Preparing a Channel as a Redundant-Pair Link, on page 3-37), the above traffic outage causes the backup peer to reboot. In most cases, this reboot has no effect on client traffic.

From cfg-channel mode, use no lacp to disable LACP:

```
no lacp
```

The CLI prompts you with a warning that this results in a brief traffic outage; enter **yes** to continue.

For example, the following command sequence disables LACP for channel 1:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# no lacp
Warning: Disabling LACP will result in temporarily loss of network
connectivity for all members of this channel.

Are you sure? [yes/no] yes
bstnA(cfg-channel[1])# ...
```

Changing the Load-Balancing Method

The channel balances the outbound load between its member ports. It chooses a port for each outbound packet by hashing the packet's source and destination IP addresses with an exclusive-OR (XOR) operation:

source-ip XOR *destination-ip*

Maximum load-balancing is achieved by using both addresses in the calculation.

Your installation may require a more-limited hash, based solely on the source or destination IP. From cfg-channel mode, use the load-balance command to reset the hash type:

```
load-balance {src-ip | dst-ip | src-dst-ip}
```

where **src-ip** | **dst-ip** | **src-dst-ip** chooses the hash type:

src-ip uses only the source-IP address in the hash. Note the ARX has a limited set of source IPs for its outbound packets: Virtual-IP addresses (*VIPs*, described later) for clients, proxy-IP addresses (also described later) for back-end filers, and management-IP addresses. This limits the distribution of traffic.

dst-ip uses only the destination address from an outbound packet. Destination IPs include all clients, back-end filers, and management stations, so they are likely to produce better distribution than a pure **src-ip** hash. However, packets to any given host always go out the same port; a heavily-used filer or client can therefore burden one port in the channel.

src-dst-ip combines the source and destination addresses with an XOR, as described above. In most installations, this produces the best traffic distribution.

This command stops and restarts the channel briefly as it resets the load-balancing hash. A prompt appears to confirm that this is acceptable; enter **yes** to continue.

For example, the following command sequence resets a channel's hash so that it only uses destination-IP addresses:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# load-balance dst-ip
Changing the load-balancing algorithm will cause the channel to disrupt traffic.

Are you sure? [yes/no] yes
bstnA(cfg-channel[1])# ...
```

Showing the Hash Results

You can show the slot and port used for a given source and destination IP. This shows the channel member that is chosen for any packet traveling from IP A to IP B. Use the show load-balancing command from any mode:

show load-balancing source-ip *source* destination-ip *dest* channel *ch-number*

where

source is a source-IP address,

dest is a destination-IP address, and

ch-number is the channel to test.

For example, the following command sequence shows that packets from 172.16.100.18 to 192.168.25.62 use port 2/5, while packets from the same source IP to 192.168.25.23 use port 2/6:

```
bstnA(cfg)# show load-balancing source-ip 172.16.100.18 destination-ip 192.168.25.62
channel 1

Report for source-ip 172.16.100.18 and destination Ip 192.168.25.62
Channel Id :1
Slot Id    :2
Interface  :5
```

```
bstnA(cfg)# show load-balancing source-ip 172.16.100.18 destination-ip 192.168.25.23
channel 1

Report for source-ip 172.16.100.18 and destination Ip 192.168.25.23
Channel Id :1
Slot Id    :2
Interface  :6
bstnA(cfg)# ...
```

Reverting to the Default Hash

As mentioned above, the **src-dst-ip** option chooses the default hash, the source-IP address XORed with the destination-IP address. This generally produces the best distribution of traffic amongst the channel's ports. The no form of the command also returns the channel to this default:

```
no load-balance
```

As with the affirmative form of the command, the CLI requests confirmation before stopping the channel and changing the hash.

For example:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# no load-balance
Changing the load-balancing algorithm will cause the channel to disrupt traffic.

Are you sure? [yes/no] yes
bstnA(cfg-channel[1])# ...
```

Setting a Channel Description (optional)

A channel description is an optional label that appears in the show channel output. From cfg-channel mode, use the description command to add a description to the current channel configuration:

```
description channel-description
```

where ***channel-description*** can have up to 15 characters. Quote the string if it has any spaces.

For example, the following command sequence assigns the description, "client 23," to channel 1:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# description "client 23"
bstnA(cfg-channel[1])# ...
```

Removing the Channel Description

From cfg-channel mode, use no description to remove the description from the current channel configuration:

```
no description
```

For example:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# no description
bstnA(cfg-channel[1])# ...
```

Enabling SNMP Traps from the Channel (optional)

A channel can issue SNMP traps for link up/down events. These traps apply to the channel as a whole, as opposed to its individual port members. From cfg-channel mode, use the no trap shutdown command to enable SNMP traps from the current channel:

```
no trap shutdown
```

For example:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# no trap shutdown
bstnA(cfg-channel[1])# ...
```

Disabling SNMP Traps

From cfg-channel mode, use the trap shutdown command to stop issuing SNMP traps for the current channel:

```
trap shutdown
```

For example:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# trap shutdown
bstnA(cfg-channel[1])# ...
```

Connecting the Member Ports

After you configure the channel at the peer station, connect the member ports on the ARX to their corresponding ports at the other station. This starts the channel operation.

Showing Channel Configuration

Use the show channel summary command to see a list of all channels:

```
show channel summary
```

This displays a table of all channels on the switch, one row per channel. Each channel is summarized in its row.

For example:

```
prtlndA(cfg)# show channel summary

Ch  Admin    Oper   Speed   Load-Balance Description
Id  State    Status         Algorithm
--- -------- ------ ------- ------------ ----------------------------
1   Enabled  Up     1  Gb/s src-dst-ip   Client/Serv CH
2*  Enabled  Up     1  Gb/s src-dst-ip   default

* Redundant-Interface

prtlndA(cfg)#
```

Showing the Load-Balancing Hash

Use the load-balance option in show channel to see the load-balancing parameters set for all the switch's channels:

show channel load-balance

For example:

```
prtlndA(cfg)# show channel load-balance

Channel Id  Load-Balance Algorithm
----------  -------------------------
1           Source and Destination Ip
2           Source and Destination Ip

prtlndA(cfg)#
```

Showing Details for One Channel

Identify a particular channel to see its detailed configuration, and some spanning-tree statistics:

show channel *id*

where ***id*** (1-8) identifies the channel.

For example, the following command shows channel 1 on a switch named "prtlndA:"

```
prtlndA(cfg)# show channel 1

Channel ID                    : 1
Name                          : Client/Serv CH
Load Balancing Algorithm      : Source and Destination Ip
Members(Slot/Interface)       : 1/3,1/4,1/5,1/6
Number of Members             : 4
Admin State                   : Enabled
Channel Oper Status           : Up
Trap Status                   : Enabled
Spanning-Tree Forwarding State: Manual Forwarding
Spanning-Tree State           : Disabled
Spanning-Tree Role            : Disabled
Accept Frames                 : All
Total VLANs Configured        : 2


Members  Tag
VLAN ID  VLAN ID
-----------------

1         N/A
98        N/A



Slot/Port    Link Status
------------------------
1/3            Up
1/4            Up
1/5            Up
1/6            Up
```

```
Spanning Tree Statistics
------------------------

STP BPDU Transmitted    0
STP BPDU Received       0
RST BPDU Transmitted    0
RST BPDU Received       0
MST BPDU Transmitted    0
MST BPDU Received       0
prtlndA(cfg)# ...
```

Showing One Channel's Statistics

Add the stats keyword to show channel id to see traffic statistics for the channel:

show channel *id* stats

where ***id*** (1-8) identifies the channel.

For example, the following command shows statistics for channel 1:

```
prtlndA(cfg)# show channel 1 stats


Channel Id  : 1
-------------------------------------------------------------------
                                Ingress                  Egress
-------------------------------------------------------------------

Octets                       1556400460               1899957595
Total Frames                    3179109                  3090787
Unicast Frames                  3162858                  3080532
Multicast Frames                   9297                        0
Broadcast Frames                   6954                    10255
PAUSE Frames                          0                        0
If Discards                           0                        0
If Errors                             0                        0
Int Mac Errors                        0                        0
If Unknown Protocol                   0
Alignment Errors                      0
CRC Errors                            0
Single Collisions                                              0
Multiple Collisions                                            0
Late Collisions                                                0
Excessive Collisions                                           0
Frames Too Long                       0

-------------------------------------------------------------------
                                RFC 1493
-------------------------------------------------------------------

TpPortDelayExceed                     0
TpPortMTUExceed                       0
TpPortInDis                           0

-------------------------------------------------------------------
                                Ether Stats
-------------------------------------------------------------------
```

```
Packet Size:64                  1161231
            65-127              1964536
            128-255              917712
            256-511              189084
            512-1023              49683
            1024-1518           1987650
            1522-2047                 0
            2048-4095                 0
            4096-9216                 0
Multicast Packets                  9297
Broadcast Packets                 17209
Total Octets                 3456358055
Good Oversize Frames                  0
Drop Events                           0
Total Discards                        0
Undersize Packets                     0
Fragments                             0
Jabbers                               0
Total Collisions                      0
CRC+Alignment Errors                  0
prtlndA(cfg)#
```

Clearing Channel Statistics

From priv-exec mode, you can use the clear counters channel command to bring all counters back to zero on all channels:

```
clear counters channel
```

To clear only one channel, specify its ID at the end of the command:

```
clear counters channel [id]
```

where ***id*** (optional, 1-8) identifies one channel. Use the show channel summary command to find all channels on the switch.

This clears all counters for each member port in the channel, thereby also clearing the channel statistics. The CLI prompts for confirmation before resetting any counters to 0 (zero); enter **yes** to proceed.

For example, the following command exits to priv-exec mode and clears the counters for channel 1:

```
prtlndA(cfg)# exit
prtlndA# clear counters channel 1

Clear the counters for all of the interfaces associated with channel
1? [yes/no] yes
prtlndA# ...
```

Showing LACP Configuration and Status for One Channel

LACP is an optional control protocol for dynamically adding and removing member ports from active use. You can add the optional lacp keyword to the end of the command to see LACP parameters and status from both ends of the channel:

```
show channel id lacp
```

where ***id*** (1-8) identifies the channel.

This shows some high-level summary information, parameters for the channel, and parameters for each port. The channel and port parameters are divided into two columns: LOCAL (for the ARX end of the channel) and PEER (for the remote end of the channel).

For example, the following command shows the LACP parameters for channel 1 on a switch named "bstnA:"

```
bstnA(cfg)# show channel 1 lacp

Channel ID                    : 1
LACP                          : Passive
Time since last state change  : 13:25:32 05/20/2008

LACP Channel Parameters :

         Local                                  Peer
------------------------------- ------------------------------------
Admin Key:       86
Oper Key:        86                 Oper Key:        86
System Priority: 100                System Priority: 32768
System ID:       00:0a:49:17:70:40  System ID:       00:0a:49:17:72:40

LACP Port Parameters:

         Local                                  Peer
------------------------------- ------------------------------------
Slot/Port:     2/7
Admin Status:* A,T,a
Oper State:  * A,T,a,S,C,D       Oper State:  * A,T,a,S,C,D
Admin Key:     86
Oper Key:      86                Oper Key:      86
Port Priority: 100               Port Priority: 0

Slot/Port:     2/8
Admin Status:* A,T,a
Oper State:  * A,T,a,S,C,D       Oper State:  * A,T,a,S,C,D
Admin Key:     86
Oper Key:      86                Oper Key:      86
Port Priority: 100               Port Priority: 0

Slot/Port:     2/9
Admin Status:* A,T,a
Oper State:  * A,T,a,S,C,D       Oper State:  * A,T,a,S,C,D
Admin Key:     86
Oper Key:      86                Oper Key:      86
Port Priority: 100               Port Priority: 0

Slot/Port:     2/10
Admin Status:* A,T,a
Oper State:  * A,T,a,S,C,D       Oper State:  * A,T,a,S,C,D
Admin Key:     86
Oper Key:      86                Oper Key:      86
Port Priority: 100               Port Priority: 0


* A - Active,        a - Aggregating, C - Collecting,   D - Distributing,
  d - Defaulted,     E - Expired,     L - Long Timeout, P - Passive,
  T - Short Timeout, S - Synchronizing

bstnA(cfg)# ...
```

Showing LACP Statistics for One Channel

You can also see transmit and receive statistics for LACPDUs (Link Aggregation Control Protocol Data Units) and marker packets. Add the optional lacp stats keywords to the end of the show channel command to see these LACP statistics for a particular channel:

show channel *id* lacp stats

where ***id*** (1-8) identifies the channel.

For example, the following command shows the LACP statistics for channel 1 on "bstnA:"

```
bstnA(cfg)# show channel 1 lacp stats

LACP Statistics:

S/P       LACP  Packets      Marker Response     Illegal    Unknown
        Tx         Rx           Tx       Rx         Rx         Rx
---- ---------- ---------- ---------- ---------- ---------- ----------
2/7  1030       1009       0          0          0          0
2/8  1033       1011       0          0          0          0
2/9  1028       1007       0          0          0          0
2/10 1032       1012       0          0          0          0


bstnA(cfg)# ...
```

On ARX-1500 and ARX-2500, you can use the clear counters lacp command to clear and restart the statistics counters for LACP.

Shutting Down a Channel

By default, a channel is running as soon as it is configured. All member ports participate in the channel, even if they were previously shut down. From cfg-channel mode, you can use shutdown to stop traffic over the current channel:

shutdown

For example, the following command sequence shuts down channel 4:

```
bstnA(cfg)# channel 4
bstnA(cfg-channel[4])# shutdown
bstnA(cfg-channel[4])# ...
```

Restarting a Channel

You restart all members of the channel as a group. From cfg-channel mode, use the no shutdown command to start the current channel and all of its member ports:

no shutdown

For example, the following command sequence starts channel 1:

```
bstnA(cfg)# channel 1
bstnA(cfg-channel[1])# no shutdown
bstnA(cfg-channel[1])# ...
```

Removing a Channel

Removing a channel has no effect on the channel's member ports, other than to remove them from the channel and return them to independent function. From cfg mode, use the no channel command to remove a channel:

```
no channel
```

For example, the following command sequence removes the configuration for channel 3:

```
bstnA(cfg)# no channel 3
bstnA(cfg)# ...
```

Preparing for Use in a Redundant-Pair Link

You can configure two ARXes as a redundant pair. A redundant pair has an *active* switch and a *standby* switch; if the active switch fails, the standby switch takes over. To configure two switches for redundancy, you first prepare a layer-2 connection between them. If you are not deploying redundant switches, you can skip this section. You can also skip this section if you are joining two ARX-500 switches; these have a dedicated port for the redundancy link, port 1/2.

This prepares the interfaces. The redundancy-configuration chapter, later, describes how to start redundancy-related processing.

Important

At a minimum, a one-gigabit connection is required for optimal failover performance. 100MB connections are supported, but cause an interruption in service after a failover: transferring data between the peers can take several minutes over the slower connection.

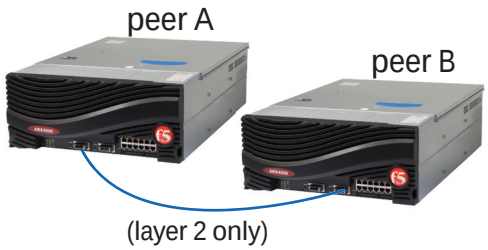
We recommend a direct connection between the redundant peers. Ideally, the switches should be co-located. A connection through a Gigabit L2 switch is permissible, provided the speed is high and the latency is low.

ARX-2000 and ARX-4000

From cfg-if-gig mode (or possibly cfg-if-ten-gig mode, on an ARX-4000), use the redundancy protocol command to prepare the current interface for use in a redundant-pair link:

```
redundancy protocol
```

For example, consider a redundant pair named Peer A and Peer B. They are cabled together from port 2/1 (Peer A) to port 2/2 (Peer B).



The following command sequences prepare the interfaces:

Peer A (port 2/1):

```
bstnA(cfg)# interface ten-gigabit 2/1
bstnA(cfg-if-ten-gig[2/1])# redundancy protocol
bstnA(cfg-if-ten-gig[2/1])# exit
bstnA(cfg)# ...
```

Peer B (port 2/2):

```
bstnB(cfg)# interface ten-gigabit 2/2
```

```
bstnB(cfg-if-ten-gig[2/2])# redundancy protocol
bstnB(cfg-if-ten-gig[2/2])# no shutdown
bstnB(cfg-if-ten-gig[2/2])# exit
bstnB(cfg)# ...
```

Removing Redundancy-Protocol Support

Use no redundancy protocol to stop the current interface from supporting a redundant-pair link:

```
no redundancy protocol
```

Important

If redundancy is established through this link, this command causes the redundant peer to reboot.

The CLI issues a warning about the peer reboot; enter **yes** to continue. If no peer is connected, this command has no effect.

For example, this command sequence stops using interface 2/1 for the redundancy protocol:

```
bstnA(cfg)# interface ten-gigabit 2/1
bstnA(cfg-if-ten-gig[2/1])# no redundancy protocol
Removing redundancy protocol for this interface will cause the peer to reboot.

Are you sure? [yes/no] yes
bstnA(cfg-if-ten-gig[2/1])# exit
bstnA(cfg)# ...
```

ARX-1500 and ARX-2500

The ARX-1500 and ARX-2500 require the configuration of a common VLAN for use as a redundant-pair link, and for exchanging heartbeat and metalog data. Prepare that VLAN for redundancy using the redundancy command in cfg-interface-vlan mode, executing the command on both chassis in the redundant pair:

Peer A:

```
bstnA(cfg-if-vlan[405])# redundancy
```

Peer B:

```
bstnB(cfg-if-vlan[405])# redundancy
```

Removing Redundancy Support

Use no redundancy to stop the current VLAN interface from supporting a redundant-pair link:

```
no redundancy
```

Important

If redundancy is established through this link, this command causes the redundant peer to reboot.

The CLI issues a warning about the peer reboot; enter **yes** to continue. If no peer is connected, this command has no effect.

Showing the Redundancy Network

The show redundancy network command shows all ports and their roles in the redundant-pair link. You can use this command once you have at least one interface configured for the redundancy protocol:

```
show redundancy network
```

This shows a table with one row per port, showing the network type (External, Private, or Metalog; the redundant-pair link carries the Private and Metalog networks), whether or not the port is enabled, whether or not the port is functioning, and the port's status in the spanning tree. Below the table of ports is a smaller table showing the count of redundant-pair-link transitions (between "up" and "down") and some details about the most recent transition.

For example, the following system has a fully-configured redundant-pair link at port 1/12:

```
prtlndA(cfg)# show redundancy network


Network  VLAN  Port(s)   Admin    Link   Spanning-Tree
                         State    Status Status
-------- ----- --------- -------- ------ -----------------
External 1     1/1       Enabled  Down   Disabled
External 1     1/2       Enabled  Down   Disabled
External 1     1/3       Enabled  Down   Disabled
External 1     1/4       Enabled  Down   Disabled
External 1     1/5       Enabled  Up     Manual Forwarding
External 1     1/6       Enabled  Up     Manual Forwarding
External 1     1/7       Disabled Down   Disabled
External 1     1/8       Disabled Down   Disabled
External 1     1/9       Disabled Down   Disabled
External 1     1/10      Disabled Down   Disabled
External 1     1/11      Disabled Down   Disabled
External 74    1/5       Enabled  Up     Manual Forwarding
External 74    1/6       Enabled  Up     Manual Forwarding
Private  1008  1/12      Enabled  Up     Manual Forwarding
Metalog  1009  1/12      Enabled  Up     Manual Forwarding


Link Transitions:
  Count:          3
  Last:           07:38:58 03/05/2010
  Reason:         Port 1/12 link up
  Last Cleared:   Never
prtlndA(cfg)# ...
```

Clearing the Link-Transition Counters

From priv-exec mode, you can use clear counters redundancy network to clear the link-transition counters:

```
clear counters redundancy network
```

For example:

```
bstnA(cfg)# end
bstnA# clear counters redundancy network
bstnA# ...
```

4

Configuring the Network Layer

Before You Begin

Use this chapter to configure the network-layer features of the ARX.

The default layer-2 parameters are sufficient for some installations, but you should verify the defaults against your environment. See *Appendix 3, Configuring Layer 2*, for a list of layer-2 defaults and the procedures to change them.

Note

The ARX-VE is a software-only virtual appliance, and, as such, functionality related to physical network infrastructure is not relevant to its operation. This includes the use of out-of-band management interfaces and VLANs.

Concepts and Terminology

A *client subnet* is an IP subnet that clients use to access services on the ARX. The switch can support multiple client subnets. These are the subnets for *Virtual-IP* (VIP) addresses, the specific addresses for accessing front-end services.

The *proxy-IP subnet* is an IP subnet that the ARX uses to connect to back-end filers and servers. This is the subnet for the switch's *proxy-IP* addresses; the switch's network processors are visible to the back-end servers through their proxy IPs. The switch supports one proxy-IP subnet.

The client subnet(s) can be the same as the proxy-IP subnet, overlap with it, or the subnets can be entirely disjoint.

The *private subnet* is internal to the ARX. This subnet must run over its own VLAN, separate from any external traffic; by default, it runs over VLAN 1002.

ASM processors also use a *metalog* VLAN to send important database logs to a battery-backed NVRAM device on the SCM. This is another private VLAN, with its own distinct subnet. The processors use IP multicast over this VLAN for all metalog data. This must be distinct from the client/server VLAN(s) as well as the private VLAN; by default, it runs over VLAN 1003.

You can join one or more ARXes together in a Resilient-Overlay Network (*RON*). Switches can exchange file replicas over the RON.

Configuring an In-Band (VLAN) Management Interface

To prepare for network-layer configuration, you must configure one in-band management interface, preferably for the VLAN with the switch's default gateway. You can configure an in-band-management interface for every VLAN on the system, if you wish, but only one is required.

From cfg mode, use the interface vlan command to create a VLAN-management interface:

interface vlan *id*

> ***id*** (1-4096) identifies the VLAN. Use the show vlan summary command for a list of configured VLANs; see *Listing all VLANs*, on page 3-9.

This places you in cfg-if-vlan mode, where you must set the IP address and start the interface. You have the option to set an interface description from this mode. You can also re-use this interface for communication with other ARXes: as the initial-rendezvous interface with a redundant peer (described below) and/or as an endpoint for one or more RON tunnels (described in a later chapter).

For example, the following command creates an in-band-management interface for VLAN 25:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ...
```

Setting the Management IP Address

The next step in configuring an in-band management interface is to set its IP address. From cfg-if-vlan mode, use the ip address command to enter an IP address for the current interface:

ip address *ip-address subnet-mask*

> where
>
> ***ip-address*** is the address of the interface (for example, 192.168.201.11), and
>
> ***subnet-mask*** defines the network-part of the address (for example, 255.255.255.0).

For example, the following command sequence sets the IP address to 192.168.25.5 for VLAN 25's in-band management interface:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ip address 192.168.25.5 255.255.255.0
bstnA(cfg-if-vlan[25])# ...
```

Setting a Description (optional)

You can optionally set a description for the in-band management interface. The description appears in the show commands. From cfg-if-vlan mode, use the description command to set a description:

description *description*

where ***description*** (1-128 characters) is your text string to describe the in-band management interface in show commands. Quote the string if it contains any spaces.

For example:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# description "management from VLAN 25"
bstnA(cfg-if-vlan[25])# ...
```

Removing the Description

From cfg-if-vlan mode, use no description to remove the description:

no description

For example:

```
bstnA(cfg)# interface vlan 6
bstnA(cfg-if-vlan[6])# no description
bstnA(cfg-if-vlan[6])# ...
```

Designating for Redundancy (optional)

You can join a pair of ARXes in a redundant pair, where one takes over the services of the other on failure. When two switches initially join as a pair, they must exchange configuration information in a *rendezvous*. The peers exchange the rendezvous information over their management interfaces, configured later as part of redundancy configuration. You will choose one management interface at each peer as the rendezvous interface. Each peer's rendezvous interface can be the out-of-band MGMT interface or any in-band interface.

◆ Note

A management interface cannot change its IP address if it is used as a rendezvous interface.

To prepare the current interface to be used later for redundant-pair rendezvous, use the redundancy command from cfg-if-vlan mode:

redundancy

For example:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# redundancy
bstnA(cfg-if-vlan[25])# ...
```

Removing the Redundancy Designation

You cannot change the redundancy designation in a pair that has already joined (described in the redundancy chapter, below). You can remove it before redundancy has been configured. From cfg-if-vlan mode, use no redundancy to remove the redundancy designation for this interface:

```
no redundancy
```

For example:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# no redundancy
bstnA(cfg-if-vlan[25])# ...
```

Enabling the Interface

The final step in configuring an in-band interface is to enable it. From cfg-if-vlan mode, use no shutdown to start the in-band management interface:

```
no shutdown
```

For example:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# no shutdown
bstnA(cfg-if-vlan[25])# ...
```

Disabling the Interface

You can shut down in-band management from a VLAN by shutting down its in-band interface. This immediately stops all connections to the interface. From cfg-if-vlan mode, use the shutdown command to disable the interface:

```
shutdown
```

For example, the following command sequence shuts down in-band management for VLAN 33:

```
bstnA(cfg)# interface vlan 33
bstnA(cfg-if-vlan[33])# shutdown
bstnA(cfg-if-vlan[33])# ...
```

Listing all VLAN Management Interfaces

From any mode, use the show interface vlan command to list all of the in-band (VLAN) management interfaces:

```
show interface vlan
```

For example:

```
bstnA(cfg)# show interface vlan

 Vlan   Admin      IP Address        Subnet Mask       Description
 ----   --------   ---------------   ---------------   -------------------
  25    Enabled    192.168.25.5      255.255.255.0

bstnA(cfg)# ...
```

Adding a Range of Proxy-IP Addresses

The next step in network-layer configuration is to establish the ARX's proxy-IP addresses. These addresses all belong to a single subnet, which has a layer-2 connection to the servers and filers. All back-end filers must belong to the same subnet as the proxy-IP addresses *or* be reachable through a static route (described below).

Each network processor requires one proxy IP, so the number of proxy IPs is platform-dependent:

- ARX-500 requires 1
- ARX-1500 requires 2
- ARX-2000 requires 4
- ARX-2500 requires 3 (or 4, depending on how the resource-profile CLI command is set)
- ARX-4000 requires 12
- ARX-VE requires 1

You enter proxy-IP addresses in ranges. The ARX automatically assigns the addresses to the various internal processors: the first number is assigned to the lowest NSM slot and processor, and subsequent proxy IPs are assigned in ascending order (for example, 2.1, 2.2, ... 2.12).

From cfg mode, use the ip proxy-address command to add one range of Proxy-IP addresses:

`ip proxy-address` ***`start-ip mask`*** **`[vlan`** ***`vlan-id`*****`] [count`** ***`number-of-ips`*****`]`**

where

start-ip is the starting address (for example, 192.168.222.110),

mask identifies the subnet-part of the ***start-ip*** (for example, 255.255.255.0),

vlan ***vlan-id*** (optional, 1-65535) is the VLAN for the proxy address(es) (use show vlan summary for a list of VLANs), and

count ***number-of-ips*** (optional, 1-64) defines the address range. The default is 1.

You can enter multiple address ranges in the same subnet and VLAN.

Note

Only one subnet is supported for the proxy-IP addresses.

Important

Be sure to assign the correct proxy-IP addresses the first time. Once the proxy IP is assigned to an NSM port, it is difficult to change. To change an assigned proxy IP, you must save your configuration (with the priv-exec

copy startup-config *command), remove it from the switch (*delete startup-config*), reboot (*reload*), edit the saved configuration with the correct proxy-IP addresses, and replay it (that is, copy it and paste it into the CLI).*

For example, the following command sequence uses two proxy-IP ranges, 192.168.25.31-34 and 192.168.25.141-148, on VLAN 25:

```
bstnA(cfg)# ip proxy-address 192.168.25.31 255.255.255.0 vlan 25 count 4
bstnA(cfg)# ip proxy-address 192.168.25.141 255.255.255.0 vlan 25 count 8
bstnA(cfg)# ...
```

Showing all Proxy IPs

Use the show ip proxy-addresses command to show all configured proxy IPs:

show ip proxy-addresses

For each proxy-IP address, this shows the MAC address, the ARX that owns it and the switch that is currently using it (relevant in a redundant pair; proxy-IPs can failover from one redundant peer to the other), and the processor that uses it.

For example:

```
bstnA(cfg)# show ip proxy-addresses

   Proxy Address    VLAN     Mac Address       Owner     In Use By    Proc
-----------------  ----  -----------------   -------   -----------   ----
   192.168.25.31/24  25   00:0a:49:17:78:80    bstnA       bstnA       2.1
   192.168.25.32/24  25   00:0a:49:17:78:81    bstnA       bstnA       2.2
   192.168.25.33/24  25   00:0a:49:17:78:82    bstnA       bstnA       2.3
   192.168.25.34/24  25   00:0a:49:17:78:83    bstnA       bstnA       2.4
   192.168.25.141/24 25   00:0a:49:17:78:84    bstnA       bstnA       2.5
   192.168.25.142/24 25   00:0a:49:17:78:85    bstnA       bstnA       2.6
   192.168.25.143/24 25   00:0a:49:17:78:86    bstnA       bstnA       2.7
   192.168.25.144/24 25   00:0a:49:17:78:87    bstnA       bstnA       2.8
   192.168.25.145/24 25   00:0a:49:17:78:88    bstnA       bstnA       2.9
   192.168.25.146/24 25   00:0a:49:17:78:89    bstnA       bstnA       2.10
   192.168.25.147/24 25   00:0a:49:17:78:8a    bstnA       bstnA       2.11
   192.168.25.148/24 25   00:0a:49:17:78:8b    bstnA       bstnA       2.12
bstnA(cfg)# ...
```

Removing a Proxy IP Addresses

Use the no form of the ip proxy-address command to remove a Proxy IP:

no ip proxy-address *proxy-ip*

where ***proxy-ip*** is the address to remove (for example, 192.168.222.110).

◆ Note

The CLI prevents you from bringing the number of proxy IPs below the required number for your system. An ARX-4000 requires 12, an ARX-2000 or ARX-2500 requires 4, an ARX-1500 requires 2, and an ARX-500 or ARX-VE requires 1.

You can also remove a proxy IP from the middle of a range, splitting the range in two.

For example, the following command sequence creates a large proxy-IP range, 192.168.25.65-84, and then removes 192.168.25.71 from the middle:

```
bstnA(cfg)# ip proxy-address 192.168.25.65 255.255.255.0 vlan 25 count 20
bstnA(cfg)# no ip proxy-address 192.168.25.71
bstnA(cfg)# ...
```

Adding a Static Route

You can use static routes to define default paths for outbound IP traffic. If an IP packet is bound for an address outside the proxy-IP subnet (above) or any client subnet, the switch uses static routes to find the correct gateway for the packet.

◆ Note

All filers must be reachable through the proxy-IP subnet or through a static route that uses a gateway on that subnet.
All clients must be reachable through any client subnet, or through a static route that uses a gateway on any client subnet.

Use the ip route command to create a static route:

ip route *ip-subnet ip-mask gateway* [*distance*]

where

ip-subnet is the address of the subnet (for example, 172.16.56.0),

ip-mask defines the network-part of the subnet (for example, 255.255.255.0),

gateway is the IP address of the forwarding router, and

distance (optional; 1-255) is the user-defined distance to the forwarding router. This metric is used for choosing between two static routes that match a desired host address; a lower distance is preferred over a higher one. The default is 128.

For example, the following command sequence creates a static route to the subnet, 172.16.100.0. The gateway for the subnet is at 192.168.25.4. The distance is set very low, so this route is likely to be chosen for all packets to this subnet:

```
bstnA(cfg)# ip route 172.16.100.0 255.255.255.0 192.168.25.4 1
bstnA(cfg)# ...
```

Setting the Default Gateway

Create a static route to 0.0.0.0 to establish a default gateway. Use the ip route command to create the static route:

ip route 0.0.0.0 0.0.0.0 *default-gateway* [*distance*]

where

default-gateway is the IP address of the default gateway, and

distance (optional; 1-255) is user-defined distance to the forwarding router. This metric is used for choosing between two static routes that match a desired host address; a lower distance is preferred over a higher one. For a default gateway, you should set this high. If you omit this, the distance is set to 128 by default.

For example, the following command sequence sets the default gateway to 192.168.25.1:

```
bstnA(cfg)# ip route 0.0.0.0 0.0.0.0 192.168.25.1
bstnA(cfg)# ...
```

Listing All Static Routes

Use the show ip route command to dump all of the ARX's static-routes:

```
show ip route
```

For example:

```
bstnA(cfg)# show ip route

Destination/Mask    Gateway          Cost Interface        Age
-------------------------------------------------------------------
0.0.0.0/0           192.168.25.1     128  VLAN25           1594
0.0.0.0/0           10.1.1.1         128  Mgmt             1575
192.168.25.0/24     0.0.0.0          0    VLAN25           Direct
10.1.1.0/24         0.0.0.0          0    Mgmt             Direct
```

Showing the Status of Static Routes

You can use the monitor keyword to see the current state of each route. This shows if the route is up or down. If the route is down, this shows the reason. If the route is up, this shows whether or not it is being used as the "Current Gateway" to reach the destination subnet (if multiple routes lead to the same subnet, only one of them is used).

```
show ip route monitor
```

For example, this shows that both default routes, one for a client/server VLAN and one for the out-of-band management interface, are up and in use:

```
bstnA(cfg)# show ip route monitor

Destination/Mask    Type  Gateway         Cost  Status  Details
-------------------------------------------------------------------------------
0.0.0.0/0           Mgmt  10.1.1.1        128   Up      Current Gateway
0.0.0.0/0           VLAN  192.168.25.1    128   Up      Current Gateway


bstnA(cfg)# ...
```

Showing Static Routes from One Processor

The SCM processor and each of the network processors have a separate routing table. All platforms have a single SCM processor (typically 1.1), but the number of network processors varies:

- ARX-4000 has 12 (indicated by slot.port 2.1 to 2.12),
- ARX-2500 has 8 (indicated by slot.port 2.1 to 2.8),
- ARX-2000 has 4 (indicated by slot.port 1.2 to 1.5),
- ARX-1500 has 6 (indicated by slot.port 2.1 to 2.6), and
- ARX-500 and ARX-VE both have 1 (indicated by slot.port 1.2).

You can specify a processor in the show ip route command to show the static routes for that processor:

show ip route from *slot.processor*

where

from is a required keyword,

slot (1 or 2 for the ARX-4000, 1 for the ARX-2000 or ARX-500) is the slot number of the desired module, and

processor (1-12) is the processor number. Use show processors for a complete list of processors (and their modules and slots) on your ARX.

For example, the following command shows all processors on an ARX-4000, then shows the static routes for processor 2.1:

```
bstnA(cfg)# show processors

                                          Memory (MB)
Proc  Module  State       Up Time             Total    Free    CPU 1m  CPU 5m
----  ------  ----------  ------------------  -------  ------  ------  ------
1.1   ACM     Up          0 days, 01:40:46      16030   15119       2       3
2.1   NSM     Up          0 days, 01:36:20       2650    2093       0       0
2.2   NSM     Up          0 days, 01:36:20       2650    2093       0       0
2.3   NSM     Up          0 days, 01:36:20       2650    2093       0       0
2.4   NSM     Up          0 days, 01:36:20       2650    2093       1       1
2.5   NSM     Up          0 days, 01:36:10       2650    2093       0       0
2.6   NSM     Up          0 days, 01:36:10       2650    2093       0       0
2.7   NSM     Up          0 days, 01:36:10       2650    2093       0       0
2.8   NSM     Up          0 days, 01:36:10       2650    2093       1       1
2.9   NSM     Up          0 days, 01:36:15       2650    2093       0       0
2.10  NSM     Up          0 days, 01:36:15       2650    2093       0       0
2.11  NSM     Up          0 days, 01:36:15       2650    2093       0       0
2.12  NSM     Up          0 days, 01:36:15       2650    2093       1       1
bstnA(cfg)# show ip route from 2.1

Proc Destination/Mask    Gateway          Cost Interface        Age
-------------------------------------------------------------------
2.1  0.0.0.0/0           192.168.25.1     128  VLAN25           5770
2.1  192.168.25.0/24     0.0.0.0          0    VLAN25           Direct


bstnA(cfg)# ...
```

For another example, the following command sequence performs the same basic steps on an ARX-2000:

```
prtlndA(cfg)# show processors

                                                Memory (MB)
Proc  Module  State       Up Time               Total    Free    CPU 1m  CPU 5m
----  ------  ----------  ------------------  -------  ------  ------  ------
1.1   ACM     Up          0 days, 02:30:04      11940   11280      28      27
1.2   ACM     Up          0 days, 02:23:00       2650    2069       0       0
1.3   ACM     Up          0 days, 02:23:00       2650    2069       0       0
1.4   ACM     Up          0 days, 02:23:00       2650    2069       0       0
1.5   ACM     Up         0 days, 02:23:00       2650    2069       1       1prtlndA(cfg)#
show ip route from 1.1

Proc Destination/Mask    Gateway          Cost Interface        Age
-------------------------------------------------------------------
1.1  192.168.78.0/24     0.0.0.0          128  VLAN             Direct
1.1  10.1.23.0/24        0.0.0.0          0    Mgmt             Direct
1.1  192.168.74.0/24     0.0.0.0          128  VLAN             Direct
1.1  0.0.0.0/0           10.1.23.1        128  Mgmt             9430


prtlndA(cfg)# ...
```

Showing Static Routes from All Processors

For a full list of the static routes on a per-processor basis, use the show ip route all command:

```
show ip route all
```

For example:

```
prtlndA(cfg)# show ip route all

Proc Destination/Mask    Gateway          Cost Interface        Age
-------------------------------------------------------------------
1.1  192.168.78.0/24     0.0.0.0          128  VLAN             Direct
1.1  10.1.23.0/24        0.0.0.0          0    Mgmt             Direct
1.1  192.168.74.0/24     0.0.0.0          128  VLAN             Direct
1.1  192.168.74.0/24     0.0.0.0          128  VLAN             Direct
1.1  10.1.23.0/24        0.0.0.0          0    Mgmt             Direct
1.1  0.0.0.0/0           10.1.23.1        128  Mgmt             9430
1.2  0.0.0.0/0           192.168.74.1     128  VLAN74           9199
1.2  192.168.74.0/24     0.0.0.0          0    VLAN74           Direct
1.2  192.168.78.0/24     10.1.25.1        128  VLAN74           9214
1.2  192.168.74.0/24     0.0.0.0          0    VLAN74           Direct
0.0  0.0.0.0/0           0.0.0.0          0    VLAN             Direct
0.0  0.0.0.0/0           0.0.0.0          2    VLAN             Direct
0.0  0.0.0.0/0           0.0.0.0          0    VLAN             Direct
1.3  0.0.0.0/0           192.168.74.1     128  VLAN74           9199
1.3  192.168.74.0/24     0.0.0.0          0    VLAN74           Direct
1.3  192.168.78.0/24     10.1.25.1        128  VLAN74           9214
1.3  192.168.74.0/24     0.0.0.0          0    VLAN74           Direct
0.0  0.0.0.0/0           0.0.0.0          0    VLAN             Direct
0.0  0.0.0.0/0           0.0.0.0          2    VLAN             Direct
0.0  0.0.0.0/0           0.0.0.0          0    VLAN             Direct


Proc Destination/Mask    Gateway          Cost Interface        Age
-------------------------------------------------------------------
```

```
0.0  0.0.0.0/0           0.0.0.0          0    VLAN           Direct
0.0  0.0.0.0/0           0.0.0.0          0    VLAN           Direct
1.4  0.0.0.0/0           192.168.74.1     128  VLAN74         9199
1.4  192.168.74.0/24     0.0.0.0          0    VLAN74         Direct
1.4  192.168.78.0/24     10.1.25.1        128  VLAN74         9214
1.4  192.168.74.0/24     0.0.0.0          0    VLAN74         Direct
1.5  0.0.0.0/0           192.168.74.1     128  VLAN74         9199
1.5  192.168.74.0/24     0.0.0.0          0    VLAN74         Direct
1.5  192.168.78.0/24     10.1.25.1        128  VLAN74         9214
1.5  192.168.74.0/24     0.0.0.0          0    VLAN74         Direct

prtlndA(cfg)# ...
```

Removing a Static Route

Use the no ip route command to remove a static route:

no ip route *ip-subnet ip-mask gateway*

where

ip-subnet is the address of the subnet (for example, 172.16.56.0),

ip-mask defines the network-part of the subnet (for example, 255.255.255.0), and

gateway is the IP address of the forwarding router.

This command removes the static route from all processors.

For example, the following command sequence removes a static route to the subnet, 172.16.100.0:

```
bstnA(cfg)# no ip route 172.16.100.0 255.255.255.0 192.168.25.4
bstnA(cfg)# ...
```

Configuring NTP

The ARX must use the same set of Network-Time-Protocol (NTP) servers that your filers and clients use. All filers and client machines must have their clocks tightly synchronized to support time-based storage policies, Kerberos authentication for Windows clients (which depends heavily on time stamps), or interconnection between multiple ARXes. From cfg mode, use the ntp server command to identify an NTP server in your network:

ntp server *ip-address*

where ***ip-address*** identifies an NTP server that is either on the proxy-IP subnet or reachable through a static route.

The ARX supports NTPv3 (RFC 1305) and SNTPv4 (RFC 2030). The default is SNTPv4, but the ARX automatically downgrades to NTPv3 if the NTP server negotiates for that version.

Repeat the command once for each server. You can enter up to eight. The ARX chooses from this list of servers based on the current jitter, synchronization distance, and other parameters defined in the protocol. For example, the following command sequence enters three NTP servers:

```
bstnA(cfg)# ntp server 192.168.25.209
bstnA(cfg)# ntp server 192.168.25.106
bstnA(cfg)# ntp server 192.168.25.201
bstnA(cfg)# ...
```

Removing an NTP Server from the List

Use the no ntp server command to remove a candidate from the list of NTP servers:

no ntp server *ip-address*

where ***ip-address*** identifies the NTP server to remove.

For example, the following command removes the NTP server at 192.168.25.106 from the configuration:

```
bstnA(cfg)# no ntp server 192.168.25.106
bstnA(cfg)# ...
```

Showing all Configured NTP Servers

The show ntp servers command lists all of the NTP servers that are in use, along with the protocol being used (3 or 4) and the number of seconds between NTP polls:

show ntp servers

For example:

```
bstnA(cfg)# show ntp servers

Configured NTP Servers

ID        Proto    Poll       Address
-----------------------------------------------------------------
```

```
2          4          64         192.168.25.201
1          4          64         192.168.25.209

bstnA(cfg)# ...
```

Showing NTP-Server Status

You can use the show ntp status command to see the current status of the ARX's NTP server:

```
show ntp status
```

The output shows one line per NTP server. It matches the output from the **pe[ers]** command in the Unix *ntpq* program, and the output from **ntpq -p** on Windows. The left-most character shows whether or not the server has been selected:

- ***** indicates that the server is in use as the "system peer."
- **o** shows that the server is in use, and that the time synchronization is derived from a pulse-per-second (PPS) signal.
- **+** means that this server is a valid candidate, though not currently selected.
- **#** means that this server is a valid candidate, but not among the first six peers sorted by synchronization distance.
- **.** means that this server is unused because, while perhaps valid, several other valid servers where closer.
- **<Space>** indicates that this server has been discarded.
- **X** shows that the server has been discarded as a "falseticker."
- **-** indicates that the server has been discarded as an "outlyer."

The st field shows the stratum of the server, where 1 is best, 15 is worst, and 16 is unusable.

The offset field shows the difference between the ARX time and the NTP-server time, in milliseconds. This should typically be less than 150.

The remaining fields require an intimate knowledge of NTP.

For example:

```
bstnA(cfg)# show ntp status

NTP server status

     remote           refid      st t when poll reach   delay   offset  jitter
------------------------------------------------------------------------------
+lager.wwmed     navobs1.mit.edu  2 -   14   64  377    0.640  -147.94   6.475
*dc1.wwmed.c     lager.wwmed      2 -   15   64  377    0.411  145.091  13.112
bstnA(cfg)# ...
```

Showing an IP-Address Configuration

Use the show ip address command to show the configuration for one IP address:

show ip address *ip-address*

where ***ip-address*** is any address.

This shows the role of the address if it is configured on the ARX. If not, this shows the address as "External."

For example, this command shows the configuration of an external address (that is, an address that is not configured on the switch):

```
bstnA(cfg)# show ip address 192.168.25.9


Report for 192.168.25.9
Slot ID    :3
Processor  :5
Type       :External
MAC Address:N/A
VLAN ID    :N/A
```

Changing the Out-of-Band-Management Interface (optional)

The out-of-band-management interface is configured by the boot wizard at initial system startup. This is the interface labeled "MGMT." It appears on the back panel of the ARX-500, ARX-2000, and ARX-4000. The ARX-1500 and ARX-2500 do not provide a dedicated out-of-band management interface; for these models, port 1/1 is configured by default as the management interface. (Although it is not recommended, this interface can be reconfigured on the ARX-1500 and ARX-2500 to be an ordinary gigabit interface.) This interface connects to a subnet that is used for managing the switch; it must be disjoint from the proxy-IP subnet or any client subnets.

◆ Note

The ARX-VE is a software-only virtual appliance, and, as such, functionality related to physical network infrastructure is not relevant to its operation. This includes the use of out-of-band management interfaces.

◆ Note

Clients and servers cannot and should never connect to the ARX through this management interface.

You have the option to change the speed, IP address, and/or description of this interface. You can also shut it down.

From cfg mode, use the interface mgmt command to access the out-of-band (OOB) management interface

```
interface mgmt
```

This places you in cfg-mgmt mode, where you can reset the interface speed and IP address, shut down or restart the interface, and/or set a description for the interface.

For example, the following command enters cfg-mgmt mode for the OOB-management interface.

```
bstnA(cfg)# interface mgmt
bstnA(cfg-mgmt)# ...
```

Setting Speed

By default, an OOB-management interface negotiates its speed with its peer(s) when it starts, choosing the highest speed possible for both ends of the connection. It also discovers the line type (fast Ethernet or fiber Ethernet) and duplex configuration (half or full) automatically. For situations where automatic negotiation is unreliable, you have the option to set the speed and duplex configuration manually.

The speed must be set to auto for the port to automatically detect the polarity (straight-through vs. crossover) on the cable.

From cfg-mgmt mode, use the speed command to set the interface speed, line type, and duplex configuration.

speed {auto | 100-half | 100-full | 10-half | 10-full | 1000-full}

where **{auto | 100-half | 100-full | 10-half | 10-full | 1000-full}** is a required choice:

auto is auto-negotiate (the default) 10/100 megabits-per-second (mbps), half/full duplex. Makes the port auto-negotiate with its peer. Use this setting to enable MDI/MDIX cross-over on the OOB-management port.

100-half is fast Ethernet, 100 mbps, half duplex.

100-full is fast Ethernet, 100 mbps, full duplex.

10-half is fast Ethernet, 10 mbps, half duplex.

10-full is fast Ethernet, 10 mbps, full duplex.

1000-full is Gigabit Ethernet, 1000 mbps, full duplex.

For the OOB-management interface, the following command sequence sets the fast Ethernet port speed to 10 mbps and the duplex configuration to full duplex.

```
bstnA(cfg)# interface mgmt
bstnA(cfg-mgmt)# speed 10-full
bstnA(cfg-mgmt)# ...
```

Changing the Management IP Address

As mentioned above, the installer sets the OOB-management interface's IP address during the initial boot-up. From cfg-mgmt mode, use the ip address command to change this IP address:

ip address *ip-address subnet-mask*

where

ip-address is the address of the interface (for example, 10.10.111.60).

subnet-mask defines the network-part of the address (for example, 255.255.255.0). This defines the management subnet, which must be entirely distinct from the proxy-IP subnet or any client subnet. Client subnets are subnets that connect to the front-end services of the switch (described in the *ARX® CLI Storage-Management Guide*).

For example, the following command sequence sets the IP address to 10.1.1.7 for the out-of-band management interface:

```
bstnA(cfg)# interface mgmt
bstnA(cfg-mgmt)# ip address 10.1.1.7 255.255.255.0
bstnA(cfg-mgmt)# ...
```

Setting a Description (optional)

You can optionally set a description for the out-of-band management interface. The description appears in the show commands. From cfg-mgmt mode, use the description command to set a description:

description *description*

where ***description*** (1-128 characters) is your text string to describe the out-of-band management interface in show commands. Quote the string if it contains any spaces.

For example:

```
bstnA(cfg)# interface mgmt
bstnA(cfg-mgmt)# description "management from the 10net"
bstnA(cfg-mgmt)# ...
```

Removing the Description

From cfg-mgmt mode, use no description to remove the description:

no description

For example:

```
bstnA(cfg)# interface mgmt
bstnA(cfg-mgmt)# no description
bstnA(cfg-mgmt)# ...
```

Disabling the Interface

The boot wizard enables the out-of-band management interface when you first configure the ARX. From cfg-mgmt mode, use the shutdown command to disable the interface:

shutdown

For example, the following command sequence shuts down the out-of-band management interface:

```
bstnA(cfg)# interface mgmt
bstnA(cfg-mgmt)# shutdown
bstnA(cfg-mgmt)# ...
```

Enabling the Interface

From cfg-mgmt mode, use no shutdown to start the out-of-band management interface:

no shutdown

For example:

```
bstnA(cfg)# interface mgmt
bstnA(cfg-mgmt)# no shutdown
bstnA(cfg-mgmt)# ...
```

Showing the Interface Configuration and Status

From any mode, use the show interface mgmt command to show the configuration and the current status of the out-of-band management interface:

show interface mgmt

For example:

```
bstnA(cfg)# show interface mgmt

Slot                    1
Interface               1
Description
Admin Status            Enabled
Link Status             Up
Speed                   1  Gb/s
Duplex                  Full
Auto Negotiation        Disabled
MAC Address             00:04:23:e2:bb:19
MTU Size                1500
IP Address              10.1.1.7
Subnet Mask             255.255.0.0
bstnA(cfg)# ...
```

Showing the Interface Statistics

You can add an optional stats keyword to show usage statistics for the management interface:

show interface mgmt stats

This shows frame and error counters for both ingress (inbound) and egress (outbound) traffic. For example:

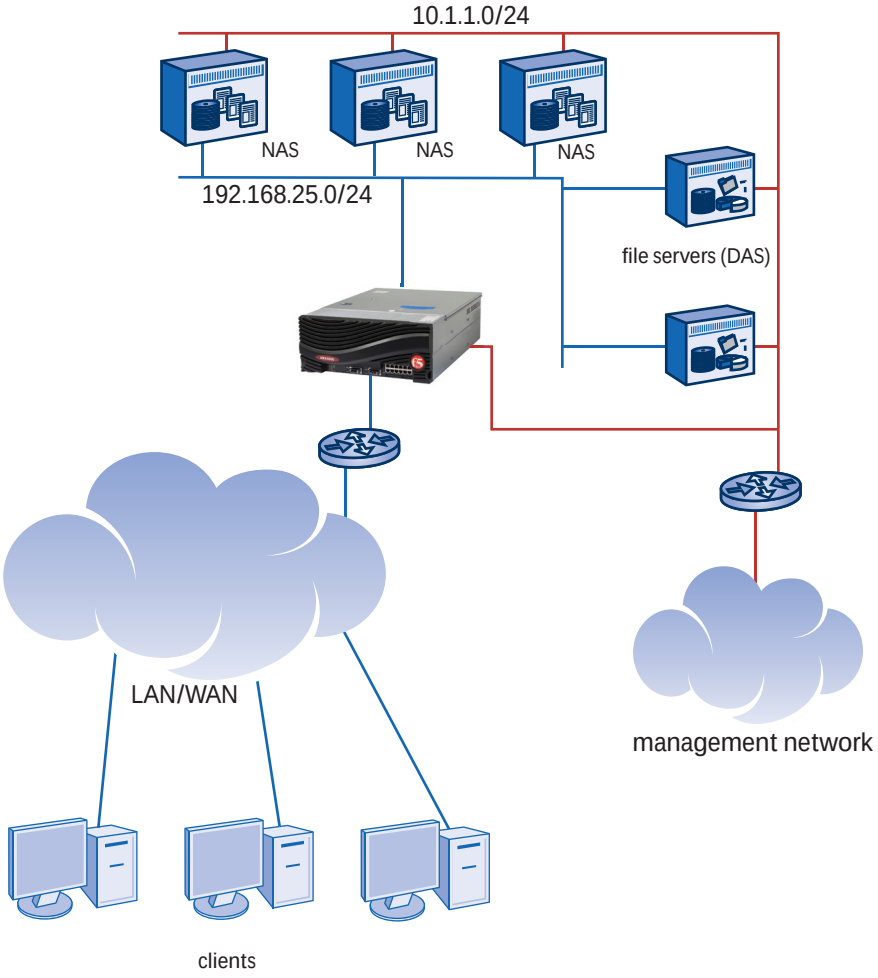
```
bstnA(cfg)# show interface mgmt stats

Slot                          1
Interface Id                  1
-------------------------------------------------------------------
                               Ingress                  Egress
-------------------------------------------------------------------

Octets                       5391453                22809824
Total Frames                   68067                   73725
Dropped Frames                     0                       0
Error Frames                       0                       0
FIFO Errors                        0                       0
Multicast Frames                  52
CRC Errors                         0
Symbol Errors                      0
Oversize Errors                    0
Frame Errors                       0
Length Errors                      0
Alignment Errors                   0
Missed Frames                      0
Collision  Frames                                          0
bstnA(cfg)# ...
```

Configuring Static Routes for Management

As mentioned above, the out-of-band management network is separate from all the data subnets. For example, the proxy-IP subnet and all client subnets might be part of the 192.168.0.0/16 network, but the management network could be on 10.1.1.0/24. The OOB interface (MGMT) therefore require a unique table of static routes, separate from the routes for the client/server subnet(s).



If an IP packet from the out-of-band management address is bound for an address outside its subnet, the ARX uses a table of management static routes to find the correct gateway for the packet. Use the mgmt flag at the end of the ip route command to create a static route in the management routing table:

```
ip route ip-subnet ip-mask gateway [distance] mgmt
```

where

ip-subnet is the address of the subnet (for example, 172.16.56.0),

ip-mask defines the network-part of the subnet (for example, 255.255.255.0),

gateway is the IP address of the forwarding router, and

distance (optional; 1-255) is user-defined distance to the forwarding router. This metric is used for choosing between two static routes that match a desired host address; a lower distance is preferred over a higher one. The default is 128.

mgmt is a required keyword to put this static route in the separate management routing table.

For example, the following command sequence creates a default gateway, 10.1.1.1, for the management network:

```
bstnA(cfg)# ip route 0.0.0.0 0.0.0.0 10.1.1.1 mgmt
bstnA(cfg)# ...
```

Listing All Management Static Routes

The show ip route command flags the management static routes with the Mgmt label under the Interface column. For example, in this command the routes in bold text are management routes:

```
bstnA(cfg)# show ip route

Destination/Mask    Gateway          Cost Interface        Age
-------------------------------------------------------------------
0.0.0.0/0           192.168.25.1     128  VLAN25           5770
0.0.0.0/8           10.1.25.1        125022 VLAN             Static
0.0.0.0/0           10.1.1.1         128  Mgmt             6185
192.168.25.0/24     0.0.0.0          0    VLAN25           Direct
192.168.25.0/24     0.0.0.0          128  VLAN             Direct
192.168.25.0/24     0.0.0.0          0    VLAN25           Direct
192.168.25.0/24     0.0.0.0          128  VLAN             Direct
192.168.78.0/24     0.0.0.0          128  VLAN             Direct
192.168.78.0/24     10.1.25.1        128  VLAN25           5789
10.1.1.0/24         0.0.0.0          0    Mgmt             Direct


bstnA(cfg)#
```

Removing a Management Static Route

Use the no ip route command with the mgmt flag to remove a static route from the management routing table:

no ip route *ip-subnet ip-mask gateway* mgmt

where

ip-subnet is the address of the subnet (for example, 172.16.56.0),

ip-mask defines the network-part of the subnet (for example, 255.255.255.0), and

gateway is the IP address of the forwarding router.

mgmt is a required keyword to remove this static route from the separate management routing table.

For example, the following command sequence removes a static route from the management table:

```
bstnA(cfg)# no ip route 192.168.33.0 255.255.255.0 10.1.1.1 mgmt
bstnA(cfg)# ...
```

Adding a Static ARP-Table Entry

The ARX uses Address-Resolution Protocol (ARP) to map IP addresses to MAC addresses. (ARP is defined in RFC 826.) Each network processor in the ARX keeps its own ARP table with its known IP/MAC mappings. You can use the arp command to create a static entry in all ARP tables:

arp *ip-address mac-address* [vlan *vlan-id*]

where

ip-address is the IP address (for example, 192.168.99.99),

mac-address is the MAC address of the above IP (for example 12:34:56:78:9a:bc), and

vlan-id (optional) is the specific VLAN for this ARP-table entry. If you enter the *vlan-id*, the ARP entry is only loaded onto modules connected to that VLAN. If you omit this option, the ARP entry is loaded onto all modules.

For example, the following command sequence creates a static ARP-table entry to map 192.168.25.1 to a MAC address:

```
bstnA(cfg)# arp 192.168.25.1 11:34:88:54:af:f5
bstnA(cfg)# ...
```

Listing ARP Entries

Every network processor in the ARX keeps a separate ARP table. Each ARP table has three flavors of IP-MAC mappings:

- *dynamic* entries learned from neighboring devices,
- *static* entries configured through the CLI (see above), and
- *local* entries, internal to the switch.

Use the show arp command to list all non-local ARP entries known to all processors:

show arp

This command provides an aggregate view of the ARP tables from all processors; duplicate entries are not shown.

For example:

```
bstnA(cfg)# show arp

IP Address         MAC Address      VLAN    Type     Age(sec)
-------------------------------------------------------------
10.1.1.1         00:01:e8:5e:ea:1f   25    dynamic    0
192.168.25.2     00:01:e8:5e:ea:1f   25    dynamic    0
```

```
10.1.11.218       00:00:00:00:00:00   25    dynamic    0
10.1.1.1          00:01:e8:5e:ea:1f   mgmt  dynamic    0
169.254.80.64     00:0a:49:17:79:46   mgmt  dynamic    0
169.254.80.65     00:0a:49:17:79:47   mgmt  dynamic    0
169.254.80.66     00:0a:49:17:79:48   mgmt  dynamic    0
169.254.80.67     00:0a:49:17:79:49   mgmt  dynamic    0
169.254.80.68     00:0a:49:17:79:44   mgmt  dynamic    0
169.254.80.69     00:0a:49:17:79:45   mgmt  dynamic    0
169.254.80.70     00:0a:49:17:79:46   mgmt  dynamic    0
169.254.80.71     00:0a:49:17:79:47   mgmt  dynamic    0
169.254.80.72     00:0a:49:17:79:48   mgmt  dynamic    0
169.254.80.73     00:0a:49:17:79:49   mgmt  dynamic    0
169.254.80.74     00:0a:49:17:79:4a   mgmt  dynamic    0
169.254.80.75     00:0a:49:17:79:4b   mgmt  dynamic    0
bstnA(cfg)# ...
```

Showing Local ARP Entries

Use show arp all to view the local (internal) ARP entries, too.

show arp all

This shows the individual ARP tables from each processor, so duplicate entries appear in the table. The processor ID (in *slot.processor* format) appears for each ARP entry. For example,

```
bstnA(cfg)# show arp all

Proc  IP Address         MAC Address      VLAN  Type     Age(sec)
-------------------------------------------------------------------
1.1  169.254.80.64    00:0a:49:17:79:46   mgmt  dynamic    0
1.1  169.254.80.67    00:0a:49:17:79:49   mgmt  dynamic    0
1.1  169.254.80.68    00:0a:49:17:79:44   mgmt  dynamic    0
1.1  169.254.80.74    00:0a:49:17:79:4a   mgmt  dynamic    0
1.1  169.254.80.75    00:0a:49:17:79:4b   mgmt  dynamic    0
1.1  169.254.80.71    00:0a:49:17:79:47   mgmt  dynamic    0
1.1  10.1.1.1         00:01:e8:5e:ea:1f   mgmt  dynamic    0
1.1  169.254.80.72    00:0a:49:17:79:48   mgmt  dynamic    0
1.1  169.254.80.70    00:0a:49:17:79:46   mgmt  dynamic    0
1.1  169.254.80.65    00:0a:49:17:79:47   mgmt  dynamic    0
1.1  169.254.80.73    00:0a:49:17:79:49   mgmt  dynamic    0
1.1  169.254.80.66    00:0a:49:17:79:48   mgmt  dynamic    0
1.1  169.254.80.69    00:0a:49:17:79:45   mgmt  dynamic    0
1.1  10.1.1.7         00:04:23:e2:9f:95   mgmt  local      -
1.1  169.254.80.32    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.33    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.76    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.77    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.78    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.79    00:0a:49:17:79:09   mgmt  local      -



Proc  IP Address         MAC Address      VLAN  Type     Age(sec)
-------------------------------------------------------------------
1.1  169.254.80.80    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.81    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.83    00:0a:49:17:79:09   mgmt  local      -
2.1  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.1  192.168.25.5     00:0a:49:17:79:ff   25    local      -
```

```
2.2  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.2  192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.3  10.1.1.1         00:01:e8:5e:ea:1f   25    dynamic    0
2.3  192.168.25.2     00:01:e8:5e:ea:1f   25    dynamic    0
2.3  10.1.11.218      00:00:00:00:00:00   25    dynamic    0
2.3  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.3  192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.4  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.4  192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.5  10.1.1.1         00:01:e8:5e:ea:1f   25    dynamic    0
2.5  192.168.25.2     00:01:e8:5e:ea:1f   25    dynamic    0
2.5  10.1.11.218      00:00:00:00:00:00   25    dynamic    0
2.5  192.168.25.15    00:0a:49:17:79:c0   25    local      -
2.5  192.168.25.12    00:0a:49:17:79:c0   25    local      -
2.5  192.168.25.141   00:0a:49:17:79:84   25    local      -



Proc  IP Address         MAC Address      VLAN  Type     Age(sec)
-------------------------------------------------------------------
2.5  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.5  192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.6  192.168.25.15    00:0a:49:17:79:c0   25    local      -
2.6  192.168.25.12    00:0a:49:17:79:c0   25    local      -
2.6  192.168.25.142   00:0a:49:17:79:85   25    local      -
2.6  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.6  192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.7  192.168.25.15    00:0a:49:17:79:c0   25    local      -
2.7  192.168.25.12    00:0a:49:17:79:c0   25    local      -
2.7  192.168.25.31    00:0a:49:17:79:80   25    local      -
2.7  192.168.25.143   00:0a:49:17:79:86   25    local      -
2.7  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.7  192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.8  192.168.25.15    00:0a:49:17:79:c0   25    local      -
2.8  192.168.25.12    00:0a:49:17:79:c0   25    local      -
2.8  192.168.25.32    00:0a:49:17:79:81   25    local      -
2.8  192.168.25.144   00:0a:49:17:79:87   25    local      -
2.8  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.8  192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.9  192.168.25.15    00:0a:49:17:79:c0   25    local      -



Proc  IP Address         MAC Address      VLAN  Type     Age(sec)
-------------------------------------------------------------------
2.9  192.168.25.12    00:0a:49:17:79:c0   25    local      -
2.9  192.168.25.33    00:0a:49:17:79:82   25    local      -
2.9  192.168.25.145   00:0a:49:17:79:88   25    local      -
2.9  10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.9  192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.10 192.168.25.15    00:0a:49:17:79:c0   25    local      -
2.10 192.168.25.12    00:0a:49:17:79:c0   25    local      -
2.10 192.168.25.34    00:0a:49:17:79:83   25    local      -
2.10 192.168.25.146   00:0a:49:17:79:89   25    local      -
2.10 10.1.11.253      00:0a:49:17:79:fe   25    local      -
2.10 192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.11 192.168.25.15    00:0a:49:17:79:c0   25    local      -
2.11 192.168.25.12    00:0a:49:17:79:c0   25    local      -
2.11 192.168.25.147   00:0a:49:17:79:8a   25    local      -
2.11 10.1.11.253      00:0a:49:17:79:fe   25    local      -
```

```
2.11 192.168.25.5     00:0a:49:17:79:ff   25    local      -
2.12 192.168.25.15    00:0a:49:17:79:c0   25    local      -
2.12 192.168.25.12    00:0a:49:17:79:c0   25    local      -
2.12 192.168.25.148   00:0a:49:17:79:8b   25    local      -
2.12 10.1.11.253      00:0a:49:17:79:fe   25    local      -



Proc  IP Address         MAC Address      VLAN  Type     Age(sec)
-------------------------------------------------------------------
2.12 192.168.25.5     00:0a:49:17:79:ff   25    local      -


bstnA(cfg)# ...
```

Showing ARP Entries From One Processor

The ARX-4000 platform has 13 processors: 1 (with multiple cores) for adaptive services and system control, and 12 for network processing. The ARX-2500 has 8 processors, the ARX-1500 has 6 processors, the ARX-2000 has 5 processors, and the ARX-500 and ARX-VE have 2 processors.

Use show processors to find the number of processors in the current switch.

Use the from clause with the show arp command to focus on one processor:

show arp from *slot.processor*

where

from is a required keyword,

slot (1-2 in the ARX-4000, 1 in other platforms) is the slot number of the desired module, and

processor (1-6 in the ARX-1500; 1-12 in the ARX-4000; 1-8 in the ARX-2500; 1-5 in the ARX-2000, or 1-2 in the ARX-500 and ARX-VE) is the processor number.

For example, the following command shows all processors on an ARX-2000, then shows the ARP table for processor 1.1:

```
prtlndA(cfg)# show processors

Proc     Module  State       Up Time             Total Kb  Free Kb  CPU1M   CPU5M
----     ------  ----------  ------------------  --------  -------  ------  -----
1.1      ACM     Up          0 days, 02:52:05    12278324  11586920 4       3
1.2      ACM     Up          0 days, 02:45:20    2714622   2034272  0       0
1.3      ACM     Up          0 days, 02:45:21    2714622   2034272  1       2
1.4      ACM     Up          0 days, 02:45:21    2714622   2034272  0       0
1.5      ACM     Up          0 days, 02:45:21    2714622   2034272  0       0
prtlndA(cfg)# show arp from 1.1


Proc  IP Address         MAC Address      VLAN  Type     Age(sec)
-------------------------------------------------------------------
1.1  10.1.23.1        00:01:e8:5e:ea:1f   mgmt  dynamic    0
1.1  10.1.23.11       00:15:17:4e:58:d9   mgmt  local      -
```

```
prtlndA(cfg)# ...
```

Showing One Type of ARP Entry

Use the type clause with the show arp command to focus on one type of ARP entry:

show arp from *slot.processor* type {local | dynamic | static}

where

from *slot.processor* chooses the processor,

type is a required keyword, and

{local | dynamic | static} is a required choice:

- **local** entries are internal to the switch,
- **dynamic** entries are learned from neighboring devices, and
- **static** entries are configured through the CLI (see *Adding a Static ARP-Table Entry*, on page 4-24).

For example, the following command shows local-ARP entries from slot 1, processor 1:

```
bstnA(cfg)# show arp from 1.1 type local

Proc  IP Address         MAC Address      VLAN  Type     Age(sec)
-------------------------------------------------------------------
1.1  10.1.1.7         00:04:23:e2:9f:95   mgmt  local      -
1.1  169.254.80.32    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.33    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.76    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.77    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.78    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.79    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.80    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.81    00:0a:49:17:79:09   mgmt  local      -
1.1  169.254.80.83    00:0a:49:17:79:09   mgmt  local      -

bstnA(cfg)# ...
```

Sending a Gratuitous ARP

It is possible for other devices in the LAN to have stale ARP entries for the ARX's proxy IPs, VIPs, or management IPs. A *gratuitous ARP* can solve this problem: it is an unsolicited ARP entry (MAC-to-IP mapping) that is broadcast to all ARP listeners on the LAN. To send a gratuitous ARP for all locally-defined IPs, use the arp gratuitous command in priv-exec mode:

arp gratuitous

The CLI asks for confirmation before sending ARPs for all IP addresses. Enter **yes** to send the gratuitous ARP or **no** to cancel.

For example, the following command sequence exits cfg mode to priv-exec mode, then sends a gratuitous ARP:

```
bstnA(cfg)# exit
bstnA# arp gratuitous
Send a gratuitous ARP for all of this switch's IP addresses? [yes/no] yes

bstnA# ...
```

Skipping the Confirmation Prompt

Use arp gratuitous yes to send all the ARPs without any confirmation prompt:

```
arp gratuitous yes
```

For example:

```
bstnA(cfg)# exit
bstnA# arp gratuitous yes
bstnA# ...
```

Sending a Gratuitous ARP for One IP Address

To send a gratuitous ARP for one proxy IP or VIP, specify the desired IP in the arp gratuitous command:

```
arp gratuitous ip-address
```

where ***ip-address*** is one proxy IP or VIP (for example, 172.16.9.4). Use the show ip proxy-addresses command for a full list of proxy-IP addresses on the ARX; for details, refer back to *Showing all Proxy IPs*, on page 4-8.

For example, the following command sequence exits cfg mode to priv-exec mode, then sends a gratuitous ARP for the proxy IP at 192.168.25.33:

```
bstnA(cfg)# exit
bstnA# arp gratuitous 192.168.25.33
bstnA# ...
```

Clearing all Dynamic-ARP Entries

Dynamic-ARP entries are learned from neighboring devices. From priv-exec mode, use the clear arp command to clear all such entries from all ARP tables in the switch:

```
clear arp
```

For example, the following command sequence exits cfg mode to priv-exec mode, then clears all dynamic-ARP entries:

```
bstnA(cfg)# exit
bstnA# clear arp
```

Clearing Dynamic-ARP Entries from One Processor

Use the from clause with the clear arp command to clear dynamic-ARP entries from a specific processor:

clear arp from *slot.processor*

where

from is a required keyword,

slot (1-2 in the ARX-4000, 1 in other platforms) is the slot number of the desired module, and

processor (1-6 in the ARX-1500; 1-12 in the ARX-4000; 1-8 in the ARX-2500; 1-5 in the ARX-2000 and ARX-1500, or 1-2 in the ARX-500) is the processor number. Use show processors for a complete list of processors (and their modules and slots) on the ARX.

For example, the following command sequence clears the dynamic-ARP entries from processor 2.6:

```
bstnA(cfg)# exit
bstnA# clear arp from 2.6
```

Removing a Static ARP-Table Entry

Use no arp to remove an IP/MAC entry from all ARP tables:

no arp *ip-address*

where ***ip-address*** is the IP address (for example, 192.168.99.99) to remove from the ARP table.

For example, the following command sequence removes the ARP-table entry for IP address 172.16.100.56:

```
bstnA(cfg)# no arp 172.16.100.56
bstnA(cfg)# ...
```

Configuring DNS Lookups

The Domain Name Service (DNS) translates IP addresses into fully-qualified domain names (FQDNs) like "www.mycompany.com." You can configure the ARX to use your local DNS servers to perform these translations. This improves the performance of the Telnet server, which attempts a DNS lookup whenever an administrator logs into the switch.

The first step in configuring DNS lookups is to identify one or more DNS servers in your network. From cfg mode, use the ip name-server command to identify a DNS server:

ip name-server *ip-address*

where ***ip-address*** is the address of the DNS server (for example, 192.168.81.11).

You can repeat this command to enter more DNS servers; the maximum is three. The ARX uses the servers in the order that you enter them. All of these servers should have knowledge of the same set of IP addresses and domain names.

For example, the following command sequence configures three DNS servers:

```
bstnA(cfg)# ip name-server 192.168.25.201
bstnA(cfg)# ip name-server 192.168.25.202
bstnA(cfg)# ip name-server 192.168.25.209
bstnA(cfg)# ...
```

Showing DNS-Lookup Configuration

Use show ip domain to view the current configuration for DNS lookups:

show ip domain

For example:

```
bstnA(cfg)# show ip domain

DNS Server Configuration
   Domain List:
   Name Servers:   192.168.25.201 192.168.25.202 192.168.25.209
bstnA(cfg)# ...
```

Providing a Local Domain List

The final (optional) step in configuring DNS lookups is to provide one or more *local domain* names. A local domain facilitates lookups for simple domain names (for example, "fs3" instead of "fs3.wwmed.com"). The ARX appends this domain name to any simple name. For example, if you set a domain name of "mycompany.com" and the switch looks up "myfiler," the switch will query its DNS server(s) with "myfiler.mycompany.com."

For installations where multiple domains apply, you can enter an ordered search list of domains. The switch uses the domains in order, appending each domain to the name in turn and proceeding to the next domain if the lookup fails.

From cfg mode, use the ip domain-list command to set a local domain name:

ip domain-list *name*

where ***name*** (1-255 characters) is a local domain name (for example, “acopia.com”).

You can repeat this command to enter up to 6 domains. The total number of characters for all domains cannot exceed 255. The ARX uses the domains in the order that you enter them.

For example, the following command sequence puts three domains into the search list:

```
bstnA(cfg)# ip domain-list wwmed.com
bstnA(cfg)# ip domain-list medarch.org
bstnA(cfg)# ip domain-list bigorg.org
bstnA(cfg)# ...
```

Removing a Domain from the Search List

From cfg mode, use no ip domain-list to remove the domain name from the search list:

no ip domain-list *name*

where ***name*** (1-255 characters) is the domain name to remove (for example, “acopia.com”).

For example, the following command sequence removes one domain name from the search list:

```
bstnA(cfg)# show ip domain

DNS Server Configuration
   Domain List:    wwmed.com medarch.org bigorg.org
   Name Servers:   192.168.25.201 192.168.25.202 192.168.25.209
bstnA(cfg)# no ip domain-list bigorg.org
bstnA(cfg)# ...
```

Testing DNS Lookups

From any mode, use expect nslookup to test the DNS connection. This command calls the DNS server to provide the IP-address/FQDN mapping for any given machine:

expect nslookup *machine*

where ***machine*** (1-128 characters) is an IP address (such as 172.16.88.45), hostname (such as “my-pc”), or a full FQDN to identify a machine on the IP network.

For example, the following command successfully looks up an IP host:

```
bstnA(cfg)# expect nslookup bboard.wwmed.com
```

```
bboard.wwmed.com is an alias for rh1.wwmed.com.
rh1.wwmed.com has address 192.168.25.19

bstnA# ...
```

Removing a DNS Server

From cfg mode, use no ip name-server to remove a DNS server from the list:

no ip name-server *ip-address*

where ***ip-address*** is the address of the DNS server to remove (for example, 192.168.201.11).

For example, the following command sequence removes the DNS server at 192.168.25.202:

```
bstnA(cfg)# show ip domain

DNS Server Configuration
   Domain List:    wwmed.com medarch.org
   Name Servers:   192.168.25.201 192.168.25.202 192.168.25.209
bstnA(cfg)# no ip name-server 192.168.25.202
bstnA(cfg)# ...
```

Showing Summaries for All Interfaces

Use the show interface summary command to show a one-line summary for each interface on the ARX:

show interface summary

Each line begins with the interface type, then includes high-level specifics about configuration and status. The interface type is abbreviated to an acronym: mgmt (the out-of-band MGMT interface) or gbe (GigaBit Ethernet, the external ports). The line then shows the slot and port number, whether or not the administrator enabled the interface, whether or not the interface is actually up, speed parameters, and the description.

For example, the following chassis has its MGMT port up, two external ports up (2/5 and 2/6), and two external ports that are administratively disabled (2/13 and 2/14):

```
bstnA(cfg)# show interface summary

Type   Slot/ Admin    Link   Speed   Duplex  Description
       Port  State    Status
-----  ----  -------  ------ ------- ------- -----------------------------------
mgmt   1/1   Enabled  Up     1  Gb/s Full
10gbe  2/1   Enabled  Down   10 Gb/s Unknown Default
10gbe  2/2   Enabled  Down   10 Gb/s Unknown Default
gbe    2/3   Enabled  Down   1  Gb/s Unknown Default
gbe    2/4   Enabled  Down   1  Gb/s Unknown Default
gbe    2/5   Enabled  Up     1  Gb/s Full    Default
gbe    2/6   Enabled  Up     1  Gb/s Full    Default
gbe    2/7   Enabled  Down   1  Gb/s Unknown Default
gbe    2/8   Enabled  Down   1  Gb/s Unknown Default
gbe    2/9   Enabled  Down   1  Gb/s Unknown Default
gbe    2/10  Enabled  Down   1  Gb/s Unknown Default
gbe    2/11  Enabled  Down   1  Gb/s Unknown Default
gbe    2/12  Enabled  Down   1  Gb/s Unknown Default
gbe    2/13  Disabled Down   1  Gb/s Unknown Default
gbe    2/14  Disabled Down   1  Gb/s Unknown Default

bstnA(cfg)# ...
```

Showing Details for All Interfaces

You can use a single show command to view all interface configurations at once. From any mode, use the show interface command without any additional arguments:

show interface

Each interface is prefaced with its Interface Type. The details match those of the individual show interface commands described above. (Some interfaces, such as Interface Type ron, are described in a later chapter.)

For example:

```
bstnA(cfg)# show interface


Interface Type              management
    Slot                    1
```

```
    Interface               1
    Description
    Admin State             Enabled
    Link Status             Up
    Speed                   1  Gb/s
    Duplex                  Full
    Auto Negotiation        Enabled
    MAC Address     00:04:23:e2:bb:01
    MTU Size                1500
Interface Type            10-gigabit
    Slot                    2
    Interface               1
    Description             Default
    Type                    10GBASE-SR X2
    Mode                    Normal
    Admin State             Enabled
    Link Status             Down
    Speed                   10 Gb/s
    Duplex                  Unknown
    Auto Negotiation(Admin) Disabled
    Flow Control(Admin)
                Receive     Off
                Send        Off
    MAC Address             00:0a:49:17:78:32
    LACP Priority           32768
    Storm Control:Broadcast 1000 packets/sec
                  Multicast 1000 packets/sec
                  Unknown DA 1000 packets/sec
    Port VLAN ID            0
    Accept Frames           Admit All
Interface Type            10-gigabit
    Slot                    2
    Interface               2
    Description             Default
    Type                    10GBASE-SR X2
    Mode                    Normal
    Admin State             Enabled
    Link Status             Down
    Speed                   10 Gb/s
    Duplex                  Unknown
    Auto Negotiation(Admin) Disabled
    Flow Control(Admin)
                Receive     Off
                Send        Off
    MAC Address             00:0a:49:17:78:33
    LACP Priority           32768
    Storm Control:Broadcast 1000 packets/sec
                  Multicast 1000 packets/sec
                  Unknown DA 1000 packets/sec
    Port VLAN ID            0
    Accept Frames           Admit All
Interface Type       gigabit
    Slot                    2
    Interface               3
    Description             Default
    Type                    Copper
    Mode                    Normal
    Admin State             Enabled
    Link Status             Down
    Speed                   1 Gb/s
    Duplex                  Unknown
    Auto Negotiation(Admin) Enabled
```

```
    Flow Control(Admin)
                Receive     Off
                Send        Off
    MAC Address             00:0a:49:17:78:34
    LACP Priority           32768
    Storm Control:Broadcast  1000 packets/sec
                  Multicast  1000 packets/sec
                  Unknown DA 1000 packets/sec
    Port VLAN ID            0
    Accept Frames           Admit All
Interface Type       gigabit
    Slot                    2
    Interface               4
    Description             Default
    Type                    Copper
    Mode                    Normal
    Admin State             Enabled
    Link Status             Down
    Speed                   1 Gb/s
    Duplex                  Unknown
    Auto Negotiation(Admin)  Enabled
    Flow Control(Admin)
                Receive     Off
                Send        Off
    MAC Address             00:0a:49:17:78:35
    LACP Priority           32768
    Storm Control:Broadcast  1000 packets/sec
                  Multicast  1000 packets/sec
                  Unknown DA 1000 packets/sec
    Port VLAN ID            0
    Accept Frames           Admit All
Interface Type       gigabit
    Slot                    2
...
    Port VLAN ID            0
    Accept Frames           Admit All




Interface Type              vlan

 Vlan   Admin      IP Address        Subnet Mask       Description
 ----   --------   ---------------   ---------------   --------------------
  25    Enabled    192.168.25.5      255.255.0.0



bstnA(cfg)# ...
```

5

ARX Feature Licensing

Overview

As of Release 5.3.0, ARX feature functionality is controlled on each ARX by a license file that determines which features are active and which are disabled, according to the terms of the license agreement associated with that specific ARX. An ARX cannot be used without a valid license. The ARX will not be able to enter global mode, and services will not start if there is no valid license, or will stop if the license expires.

The license identifies the ARX features that are associated with a unique base registration key that is assigned to a specific ARX. In addition to the base registration key, the license also contains the ARX software version, the license start date, the license end date, the service check date, the platform ID, and the ARX's dossier fingerprint, which is an encrypted block of ARX-specific data. The license also provides a mechanism for managing the lifespan of ARX service contracts.

Before You Begin

Activating an ARX license requires the use of a base registration key and a dossier that is generated using that key.

License Types

F5 Networks provides evaluation licenses for ARX hardware platforms, trial licenses for ARX-VE software, and production licenses for all ARX platforms. Production licenses are perpetual and require a valid service contract in order to be upgraded. Evaluation licenses are valid for 30 or 45 days, as specified by the terms of the individual evaluation. Trial licenses are used with ARX-VE only, and are valid for 90 days.

Base Registration Keys and Add-On Keys

The relationship between a customer and an ARX license is tracked using a base registration key, which is an alphabetical string that is used to activate the features and products that the customer purchased for a particular ARX. Each ARX has a unique base registration key associated with it which is used during product activation. Each new ARX has its base registration key configured on it at the factory. For an older ARX unit, you must obtain the corresponding base registration key from F5 Networks.

In addition to the original base registration key associated with each ARX, add-on registration keys enable you to amend an existing license to activate additional features or increased system limits beyond those supported currently on a particular ARX. This enables you to expand your ARX's functionality on an as-needed basis. Once an add-on registration key has been activated, it is associated with the base registration key on the license server. Contact F5 Networks to obtain an add-on registration key.

Dossiers

In addition to the base registration key, ARX license validation and activation rely on the use of a dossier, which is an encrypted set of data used to identify a particular ARX device.

The ARX automatically generates and transmits the dossier during the *automatic license activation* procedure. If you use the *manual license activation* procedure, you generate and transmit the dossier explicitly during the manual activation. This is described in a later section.

Licensing Considerations For Redundant Pairs

Each ARX in a redundant pair requires its own license. Therefore, it is necessary for both ARXes in the pair to have the same licensed feature flags, or problems can occur during failover. Use the CLI command, show redundancy license, to display license information for a redundant pair.

License Activation

A license must be activated in order for it to take effect and enable the corresponding ARX functionality. License activation involves the use of an activation server in conjunction with either an automatic license activation procedure or a manual license activation procedure.

◆ Note

Evaluation licenses and ARX-VE trial licenses have built-in expiration dates, and begin counting down toward those expiration dates as soon as they are activated.
If you install an evaluation license on a redundant pair of ARX devices, the start dates should be no more than 24 hours apart. F5 recommends activating the license at both peers on the same day. This avoids a large gap in time during which one peer is licensed and the other is unlicensed, and it also avoids an SNMP trap that warns you of the license mismatch.

License Activation Server

Activation servers permit and record the activation of a license, checking to ensure that the license is being activated on the same ARX for which the customer paid for functionality. This prevents multiple activations of functionality on different hardware using the same base registration key. The ARX communicates with the licensing server over SSL.

You can use the ping license-server command to verify that the license activation server (https://activate.f5.com) is able to respond to license activation requests.

Automatic Activation Procedure

Typically, activating an ARX license is a very simple procedure, requiring only the execution of a single CLI command. This is known as automatic activation, and it can be performed on an ARX that is able to access the F5 Networks licensing server over the Internet. The activation server may require the ARX administrator to accept an end-user license agreement (EULA) or provide additional information before the license is activated. Once all of the prerequisites have been satisfied, the activation server provides a license text file, which is loaded on the ARX subsequently.

Execute the license activate CLI command to activate the ARX license in this way. The ARX must be able to communicate with the F5 Networks license activation server over SSL (TCP port 443) in order for this to work. If this is not possible (perhaps due to security policies at the ARX site), use the manual license activation procedure instead (see *Manual Activation Procedure*, on page 5-9).

ARX With Factory-configured Base Registration Key

To initiate an automatic license activation using the CLI on a new ARX that has its base registration key preconfigured, simply execute the privileged exec mode command license activate without any arguments, as shown here:

```
license activate
```

This command contacts the F5 license activation server and supplies the preconfigured base registration key automatically.

For example:

```
bstnA# license activate
% INFO: The license has been successfully activated.

bstnA#
```

Answer "yes" to accept the end user license agreement (EULA) that is returned from the activation server.

Execute the show active-license command to display the list of features and system limits governed by the license.

ARX With Acquired Base Registration Key

To initiate an automatic license activation using the CLI for an ARX for which the base registration key was not configured at the factory but was obtained from F5 Networks at a later time, type the privileged exec mode command license activate with the base-reg-key argument. The syntax is:

```
license activate base-reg-key basekey
```

where *basekey* is the base registration key of the ARX for which the license is being activated.

For example:

```
bstnA# license activate base-reg-key CRJGVQP-DYWST-ANKR-GBYYDMT
% INFO: The license has been successfully activated.

bstnA#
```

The licensing software generates the dossier and submits a licensing activation request to the activation server.

Answer "yes" to accept the end user license agreement (EULA) that is returned from the activation server.

Displaying the Active License

Execute the show active-license command to display the list of features and system limits governed by the license.

show active-license

For example:

```
bstnA# show active-license
System License Information
--------------------------

Auth Vers:                      5b
Usage:                          F5 Internal Product Development
Registration Key:               CRJGVQP-DYWST-ANKR-GBYYDMT
Licensed version:               6.0.0
License Date:                   Nov 11 2010
License Start:                  Nov 10 2010
License End:                    Mar 20 2011
Service Check Date:             Feb 18 2011
Platform ID:                    C101
Service Status:                 As of 2011-02-18 there is no active service contract.
 :                               This may inhibit your ability to upgrade your software.

License Load Date:              Feb 18 2011

Module List
-----------------
ARX 1000:
Reg Key:                        U381339-5329783


Feature List
-----------------
cifs_exports_per_system:                16000
cifs_services_per_system:               64
direct_attach_points_per_system:        65536
direct_shares_per_system:               4096
direct_shares_per_volume:               255
direct_shares_per_volume_group:         4096
files_per_system_4k:                    93750
files_per_volume_4k:                    31250
files_per_volume_group_4k:              93750
global_servers_per_system:              64
namespaces_per_system:                  4
protocol_qty_allowed:                   3
redundancy:                             enabled
shares_per_system:                      128
shares_per_volume:                      64
shares_per_volume_group:                128
virtual_services_per_system:            64
volume_groups_per_system:               4
volumes_per_system:                     64
volumes_per_volume_group:               64

bstnA#
```

Manual Activation Procedure

The manual license activation procedure is provided for cases in which the ARX in question is not able to access the F5 Networks license activation server over the Internet. The manual activation process still takes place at the F5 activation server web page, but with the significant difference that the administrator uses CLI commands to generate the dossier and load the returned license file.

Initiate a manual license activation by first generating the system dossier. Do this by executing the license create license-dossier CLI command. For example, the following command:

```
bstnA# license create license-dossier base-reg-key
CRJGVQP-DYWST-ANKR-GBYYDMT
```

generates a dossier for the ARX with the base registration key CRJGVQP-DYWST-ANKR-GBYYDMT. The name of the dossier file is "arx.dossier".

Use the command show license-dossier to display the encrypted contents of the resulting system dossier:

```
bstnA# show license-dossier
45f60855bc0fb4541ef6d56fdf5e2cf07a8e2831a7b99476e2acbb41af57a76
6c005c2a68a2a84ddaaee1733604b63284684560f0062c0784b72e01595b8fe
8c8e20987682d9c4754e07ab77b3fd5e33ac90dd5d40e03e55beb1a7b1d3ffc
b44073d5a762647a6967354cc374592f170158d252e622c8a98a1a8107b7122
4f442c08a73ae52f7b61015de48dae42c30c809dadbb73ecbf3977e063bfcbf
d09b879b6a64c40dc39e9b7ba87606...
bstnA#
```

Copy the resulting system dossier file to a desktop that has a web browser, and that is capable of accessing the F5 Networks licensing server at https://activate.f5.com. From that desktop, open the F5 activation server web page in the browser and follow the instructions there to submit the dossier.

◆ Note

Copy the dossier using your preferred file copy method, such as copy ftp*,* copy scp*,* copy smtp*, etc. Alternatively, you can copy and paste the encrypted dossier contents from the dossier file to the appropriate field in the activation server web page.*

Review and accept the EULA that is returned from the activation server.

Copy the returned license file, named "arx.license", from the desktop to the "configs" directory on the ARX, again using your preferred file copy method.

Activate the license file on the ARX using the license activate file CLI command. For example, the following command:

```
bstnA# license activate file arx.license
```

activates the license file named "arx.license".

Executing the show active-license command displays the list of features and system limits that are enabled by the license.

Displaying License Files

Executing the show license CLI command displays the list of license files that are present on the current ARX. This command, without any arguments, lists the contents of the ARX's license directory. If you specify a particular license file as an argument (e.g., show license active.license), the contents of that license file will be displayed instead.

If you are using ARXes in an HA pair, executing the show redundancy license CLI command displays the license information for the current ARX's HA peer.

Executing the clear active-license CLI command deletes the license that is active on the current ARX, rendering that ARX unlicensed.

For example:

```
bstnA# show license


  license
      active.license          Feb 17 18:11  9.8 kB
      arx.license             Jul  9  2010  3.5 kB
bstnA#
```

Activating Add-On Registration Keys

Add-on registration keys enable you to amend your existing license to activate additional features or increased system limits beyond those supported currently on your ARX. This enables you to expand your ARX's functionality on an as-needed basis. Activating an add-on key causes F5 Networks to issue you an amended license file that supersedes your previous license file. Once an add-on key has been activated, it is associated with the corresponding base registration key on the license server.

Contact F5 Networks to obtain an add-on license key.

Activate an add-on key using this command:

`license activate add-on-key <key>`

where <*key*> is the add-on registration key string.

If you are using manual license activation, execute the following command first to generate a dossier:

`license create license-dossier add-on-key <key>`

where <*key*> is the add-on registration key string.

Copy the resulting system dossier file to a desktop that has a web browser, and that is capable of accessing the F5 Networks licensing server at https://activate.f5.com. From that desktop, open the F5 activation server web page in the browser and follow the instructions there to submit the dossier.

◆ Note

Copy the dossier using your preferred file copy method, such as copy ftp*,* copy scp*,* copy smtp*, etc. Alternatively, you can copy and paste the encrypted dossier contents from the dossier file to the appropriate field in the activation server web page.*

Review and accept the EULA that is returned from the activation server.

Copy the returned license file, named "arx.license", from the desktop to the "configs" directory on the ARX, again using your preferred file copy method.

Next, activate the updated license manually:

`license activate file my.license`

where *my.license* is the name of your license file.

For example:

```
bstnA# license activate file arx.license
% INFO: The license has been successfully activated.

bstnA#
```

Each feature that is newly enabled and each system limit that is changed is announced with a message and a trap.

For example:

```
bstnA# license activate file arx.license
% INFO: The license has been successfully activated.

% INFO : feature flag cifs_services_per_system changed from '64' to '8'
% INFO : feature flag direct_attach_points_per_system changed from '131072' to '16384'
% INFO : feature flag direct_shares_per_system changed from '6144' to '2048'
% INFO : feature flag direct_shares_per_volume changed from '255' to '127'
% INFO : feature flag direct_shares_per_volume_group changed from '3072' to '2048'
% INFO : feature flag files_per_system_4k changed from '187500' to '46875'
% INFO : feature flag files_per_volume_4k changed from '31250' to '15625'
% INFO : feature flag files_per_volume_group_4k changed from '93750' to '46875'
% INFO : feature flag global_servers_per_system changed from '64' to '8'
% INFO : feature flag memory changed from '8192' to '4096'
% INFO : feature flag namespaces_per_system changed from '6' to '2'
% INFO : feature flag protocol_qty_allowed changed from '3' to '2'
% INFO : feature flag shares_per_system changed from '384' to '64'
% INFO : feature flag shares_per_volume changed from '64' to '16'
% INFO : feature flag shares_per_volume_group changed from '192' to '32'
% INFO : feature flag virtual_services_per_system changed from '64' to '8'
% INFO : feature flag volume_groups_per_system changed from '6' to '2'
% INFO : feature flag volumes_per_system changed from '96' to '32'
% INFO : feature flag volumes_per_volume_group changed from '48' to '16'
bstnA#
```

Note

If you are running in a redundant configuration, make sure that you activate the add-on key on the other member of the redundant pair, as well.

License Enforcement and Monitoring

An ARX cannot be used without a valid license. The ARX will not be able to enter global mode, and services will not start if there is no valid license, or will stop if the license expires. License enforcement will also enforce system limits specified by the license. Traps and log messages will be issued if there is no valid license, or if a valid license expires. If the service contract expires, the ARX will not allow upgrades.

The ARX provides notification when a trial license is close to expiring. Trial license expiration warnings are provided via SNMP trap, Email, and login banner when the license is 30 days, 15 days, and 7 days away from expiring. During the last 7 days that a trial license is valid, notifications are provided daily.

System Limit Enforcement

The software license specifies the set of feature and system limits for the ARX. The ARX software enforces some limits in the current release, but other limits are not enforced, although they may be enforced in a subsequent release. Limits that are not enforced currently are ignored by the ARX software.

The show active license command displays limits that are enforced by the ARX software. These enforced limits typically are associated with configured values. Limits that are not enforced are omitted from this command's output.

◆ Note

Platform limits that derive from hardware resources supersede limits specified by the software license.

Upgrading an ARX To Use License-Aware Software

An ARX running an older release of ARX software that is not license-aware (that is, that is not governed by a software license) will become license-aware when its software is upgraded to the current release.

Follow this procedure when upgrading an HA pair to use license-aware ARX software:

- Log in to the backup ARX.
- Copy the new software release to the backup ARX.
- Execute the boot system command on the backup ARX.
- Reload the ARX software on the backup ARX. When the backup ARX is running for the first time with the new software, the license must be activated. Use automatic or manual license activation, as appropriate for your installation.
- When license activation succeeds, HA pairing occurs. Wait until HA pairing has re-established between both ARXes in the cluster.
- Fail over the currently active (down-rev) ARX to the backup ARX, activating the new, license-aware software.
- When the failed ARX returns, repeat the first five steps to complete the cluster upgrade.

Note

When a user logs in to the ARX, the message of the day announces that the ARX is not licensed. An SNMP trap to this effect is sent, also.

Note

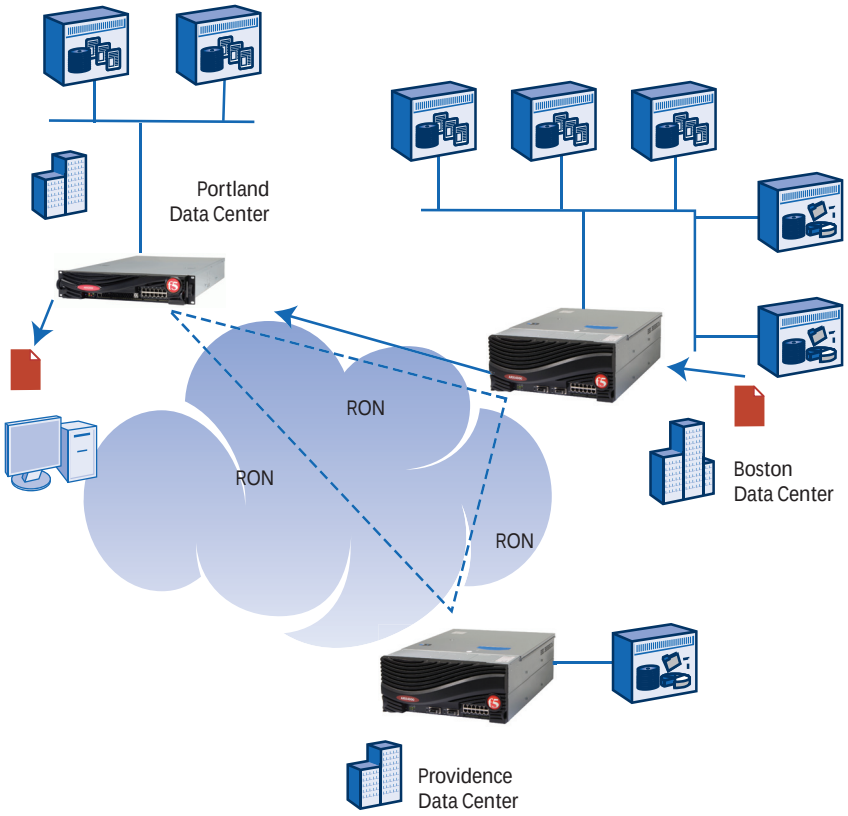
If the license activation fails and cannot be resolved, you must reload and boot the earlier release of software.When this is done, the ARX's switch configuration is restored from the pre-upgrade saved copy, which avoids re-configuring the ARX and re-importing. This also causes any changes made to the running configuration after the software upgrade to be lost.

6

Joining a RON

Overview

Use this chapter to join two or more ARXes together with a Resilient-Overlay Network (RON) of IP tunnels. Switches on a RON can exchange large sets of client files, so that clients in front of a remote switch can access read-only copies of files hosted by a local switch.



Before You Begin

Before you can establish any RON tunnels, you must have at least one in-band (VLAN) management interface. This is the endpoint for a RON tunnel. To create an in-band management interface, refer back to *Configuring an In-Band (VLAN) Management Interface*, on page 4-4.

You must also ensure that the clocks are synchronized between each switch and all of its filers. Drastic clock resets can cause serious problems for RON communication; the switches in the RON send updates to one another based on an internal timer. Configure the ARX to use the same NTP servers that the filers use, as shown in *Configuring NTP*, on page 4-15.

Creating a Tunnel to Another ARX

RON configuration involves creating a series of tunnels between ARXes. A RON tunnel terminates at an in-band (VLAN) management interface, so you create one from cfg-if-vlan mode. One in-band management interface can support multiple RON tunnels. Use the ron tunnel command to create a RON tunnel to another switch.

◆ Note

RON tunnels do not currently support encryption. Tunnels must traverse a secure IP network at this time.

ron tunnel *name*

where ***name*** (1-32 characters) is the name you choose for the RON tunnel (for example, “to-winnepeg”).

This puts you into cfg-if-vlan-ron-tnl mode, where you must set the remote IP addresses for the tunnel and then activate the interface. You can also set optional heartbeat interval and heartbeat-failure threshold from this mode.

For example, the following command sequence creates a tunnel named “toPortland” that terminates in VLAN 25:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ron tunnel toPortland
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# ...
```

Configuring the Peer IP

The next step in configuring a RON tunnel is to provide the peer-IP address for the tunnel. This is the in-band management address used at the *other* end of the tunnel; the tunnel cannot function unless each end knows the other’s IP address. From cfg-if-vlan-ron-tnl mode, use the peer address command to identify the peer’s IP address:

peer address *ip-address*

where ***ip-address*** identifies the peer address (for example, 172.200.60.3).

For example, the following command sequence sets the remote IP address for the “toPortland” tunnel:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ron tunnel toPortland
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# peer address 192.168.74.66
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# ...
```

Setting the Heartbeat Interval (optional)

The ARX periodically sends a "heartbeat" to verify connectivity and measure latency. The default heartbeat interval, 3 (seconds), is sufficient for most installations. From cfg-if-vlan-ron-tnl mode, use the heartbeat interval command to change the heartbeat interval:

heartbeat interval *seconds*

where ***seconds*** (1-30) is the number of seconds between tunnel heartbeats.

For example, the following command sequence sets the heartbeat interval to 10 seconds:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ron tunnel toPortland
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# heartbeat interval 10
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# ...
```

Setting the Heartbeat-Failure Threshold (optional)

By default, the ARX declares a tunnel "Down" after 4 consecutive heartbeat failures. From cfg-if-vlan-ron-tnl mode, use the heartbeat failure command to change this threshold for heartbeat failures:

heartbeat failure *number-heartbeats*

where ***number-heartbeats*** (2-10) is the number of consecutive, unanswered heartbeats to use as a threshold.

For example, the following command sequence sets the heartbeat-failure threshold to three:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ron tunnel toPortland
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# heartbeat failure 3
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# ...
```

Activating the Tunnel Interface

The final step in configuring a tunnel interface is to activate it. Both ends of the tunnel must be activated for the tunnel to function, and their IP/peer addresses must agree. From cfg-if-vlan-ron-tnl mode, use no shutdown to start the tunnel:

no shutdown

For example:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ron tunnel toPortland
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# no shutdown
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# ...
```

Shutting Down the Interface

From cfg-if-vlan-ron-tnl mode, use the shutdown command to shut down the RON tunnel:

shutdown

For example, the following command sequence shuts down the “toPortland” tunnel:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ron tunnel toPortland
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# shutdown
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# ...
```

Configuring the Other End of the Tunnel

At the peer switch, set the peer address and enable the tunnel. Heartbeat settings should be the same for each peer. For example, this command sequence configures a tunnel interface at the other end of the “toPortland” tunnel, in VLAN 74:

```
prtlndA(cfg)# interface vlan 74
prtlndA(cfg-if-vlan[74])# ron tunnel toBoston
prtlndA(cfg-if-vlan-ron-tnl[74~toBoston])# peer address 192.168.25.5
prtlndA(cfg-if-vlan-ron-tnl[74~toBoston])# heartbeat interval 10
prtlndA(cfg-if-vlan-ron-tnl[74~toBoston])# heartbeat failure 3
prtlndA(cfg-if-vlan-ron-tnl[74~toBoston])# no shutdown
prtlndA(cfg-if-vlan-ron-tnl[74~toBoston])# exit
prtlndA(cfg)# ...
```

Showing the RON Configuration

From any mode, use the show ron command to display the configuration and current status for the RON:

show ron

This lists every switch in the RON, with a table of status information for each.

For example, the following command shows the RON from the “prtlndA” switch:

```
prtlndA(cfg)# show ron

Switch Name             HA Peer Switch                                Uptime
Status                  UUID                                 Management Addr
-------------------------------------------------------------------------------
bstnA                   (None)                              0 days, 02:07:16
ONLINE                  d9bdece8-9866-11d8-91e3-f48e42637d58          10.1.1.7

canbyA                  (None)                              0 days, 02:09:41
ONLINE                  64a6417e-cc3d-11df-80ca-a73fbeb72ef8       10.1.33.105

newptA                  (None)                              0 days, 02:07:16
ONLINE                  cf251849-826d-01a8-9110-8dtu78fca5b2       10.1.117.74

provA                   (None)                              0 days, 02:06:07
ONLINE                  db922942-876f-11d8-9110-8dtu78fc8329        10.1.38.19

prtlndA                 prtlndB                             0 days, 02:09:04
ONLINE                  876616f6-79ac-11d8-946f-958fcb4e6e35        10.1.23.11
```

```
prtlndB                 prtlndA                                 0 days, 02:08:56
ONLINE                  64dcab94-a2b6-11d8-9d25-bf2c991c83f9        10.1.23.12

stkbrgA                 (None)                                  0 days, 02:08:36
ONLINE                  0a77ea76-ad5b-11e0-bb62-cf36a2f76cec     192.168.66.62

stoweA                  (None)                                  0 days, 02:09:39
ONLINE                  05d5a0fa-f2fb-11df-8daf-af50d57e388e        10.1.14.76
prtlndA(cfg)# ...
```

Focusing on One RON Member

To display the RON for one peer only, use the show ron command with the peer's host name:

show ron *hostname*

where ***hostname*** (1-128 characters) identifies the RON member to show.

This example focuses on the switch named prtlndA:

```
prtlndA(cfg)# show ron prtlndA

  Switch Name:           prtlndA
  HA Peer Switch:        (None)
  Status:                ONLINE
  Uptime:                0 days, 00:24:52
  UUID:                  3d17e8ce-571e-11dc-9852-ef323fbb290f

prtlndA(cfg)# ...
```

Showing All RON Tunnels

Use the show ron tunnel command to view a summary table of all RON tunnels, one row per tunnel:

show ron tunnel

Each tunnel has its own row showing the connection state, its management interface, remote address, and uptime.

For example, this shows three RON tunnels coming from the "prtlndA" switch:

```
prtlndA(cfg)# show ron tunnel

Name            State       Interface       Remote Addr     Up Time
-------------------------------------------------------------------------
toCanby         Connected   VLAN/25         192.168.121.75  0d, 01:43:24
toNewport       Connected   VLAN/25         192.168.8.106   0d, 01:43:55
toProvidence    Connected   VLAN/25         192.168.103.160 0d, 01:44:27

prtlndA(cfg)# ...
```

Showing the Details of One RON Tunnel

Identify a particular RON-tunnel name to show the full configuration for that tunnel:

show ron tunnel *name*

where ***name*** (1-32 characters) identifies the tunnel.

For example, the following command shows the full configuration of the "toBoston" interface:

```
prtlndA(cfg)# show ron tunnel toBoston

            Name: toBoston
            Peer: bstnA
    Tunnel State: Connected
          Uptime: 0d, 00:11:35
       Interface: VLAN/74
  Remote Address: 192.168.25.5
 Security Policy:
 Ping Fail Limit: 3
   Ping Interval: 10 (seconds)
 Round Trip Time: 0.1 (ms)
Packet Lost Rate: 0.59 (%)
  TCP Throughput: 125000000 (Bytes/sec)
Loss-RTT Product: 1 (us)
  RON Packets In: 173
 RON Packets Out: 101
 Data Packets In: 325770
Data Packets Out: 139252
   Data Bytes In: 431237 (KB)
  Data Bytes Out: 7459 (KB)
 Local Processor: 2.2
 Last Error Code: 0
  Control Errors: 0
     Data Errors: 0
```

Showing the Details of All RON Tunnels

Use the all keyword to show these details for all RON tunnels:

```
show ron tunnel all
```

Showing RON Tunnels For Redundant ARX-1500 and ARX-2500 Pairs

Use the show ron tunnel redundancy command to display information about the RON tunnel that is created automatically when two ARX-1500s or ARX-2500s are configured to be redundant peers. This command is available only on an ARX-1500 or ARX-2500 with a redundant peer. The command syntax is:

```
show ron tunnel redundancy
```

This RON tunnel was created automatically when the two ARX-1500s or ARX-2500s were joined as a redundant pair, and it carries redundancy-related traffic between the two peers.

Showing the RON Database

The internal RON software keeps routing information and statistics for all RON tunnels. The peers exchange Link-State Advertisement (LSA) packets through the RON tunnels, to share their view of the RON network. The LSAs are stored in a local database. To view this database of RON information, use the show ron database command:

```
show ron database
```

This shows a separate table for each switch in the RON. Each table shows the IP subnets that are relevant to the switch and a sub table with one row per RON tunnel. Each tunnel row shows the average Round-Trip Time (RTT) for packets, the percentage of lost packets, and other useful statistics.

For example:

```
prtlndA(cfg)# show ron database

Hostname          Status   Serial #    Age   Private Subnet(s)
-------------------------------------------------------------------------------
bstnA             Current  4E8936BC    870   169.254.80.0/24

  Tunnel          Peer            State      RTT(ms) Loss(%) TCP(Kb/s) Loss*RTT
  ---------------------------------------------------------------------------
  toCanby         canbyA          Connected     0.1     0.6   122070        0
  toNewport       newptA          Connected     0.1     0.6   122070        0
  toPortland      prtlndA         Connected     0.1     0.6   122070        0
  toPortlandB     prtlndB         Connected     0.1     0.6   122070        0
  toProvidence    provA           Connected     0.1     0.6   122070        0
  toStockbridge   stkbrgA         Connected     0.2     0.6   122070        1
  toStowe         stoweA          Connected     0.1     0.6   122070        1


Hostname          Status   Serial #    Age   Private Subnet(s)
-------------------------------------------------------------------------------
canbyA            Current  4E89368B    425   169.254.116.0/24

  Tunnel          Peer            State      RTT(ms) Loss(%) TCP(Kb/s) Loss*RTT
  ---------------------------------------------------------------------------
  toBoston        bstnA           Connected     0.1     0.6   122070        1


Hostname          Status   Serial #    Age   Private Subnet(s)
-------------------------------------------------------------------------------
newptA            Current  4E8936AB     85   169.254.52.128/26

  Tunnel          Peer            State      RTT(ms) Loss(%) TCP(Kb/s) Loss*RTT
  ---------------------------------------------------------------------------
  toBoston        bstnA           Connected     0.1     0.6   122070        0
  toProvidence    provA           Connected     0.1     0.6   122070        0


Hostname          Status   Serial #    Age   Private Subnet(s)
-------------------------------------------------------------------------------
provA             Current  4E8936B4    400   169.254.3.64/26

  Tunnel          Peer            State      RTT(ms) Loss(%) TCP(Kb/s) Loss*RTT
  ---------------------------------------------------------------------------
  toBoston        bstnA           Connected     0.1     0.6   122070        0
  toNewport       newptA          Connected     0.1     0.6   122070        0


Hostname          Status   Serial #    Age   Private Subnet(s)
-------------------------------------------------------------------------------
prtlndA           Current  4E8936F3    830   169.254.140.0/24   169.254.58.0/24

  Tunnel          Peer            State      RTT(ms) Loss(%) TCP(Kb/s) Loss*RTT
  ---------------------------------------------------------------------------
  Redundancy      prtlndB         Connected       0     0.6   122070        0
  toBoston        bstnA           Connected     0.1     0.6   122070        0
```

```
Hostname          Status   Serial #    Age   Private Subnet(s)
-------------------------------------------------------------------------------
prtlndB           Current  4E8936FE    865   169.254.58.0/24   169.254.140.0/24

  Tunnel          Peer            State      RTT(ms) Loss(%) TCP(Kb/s) Loss*RTT
  ---------------------------------------------------------------------------
  Redundancy      prtlndA         Connected        0     0.6   122070        0
  toBoston        bstnA           Connected      0.3     0.6   105013        2


Hostname          Status   Serial #    Age   Private Subnet(s)
-------------------------------------------------------------------------------
stkbrgA           Current  4E89368B    445    169.254.4.0/24

  Tunnel          Peer            State      RTT(ms) Loss(%) TCP(Kb/s) Loss*RTT
  ---------------------------------------------------------------------------
  toBoston        bstnA           Connected      0.3     0.6   115552        2


Hostname          Status   Serial #    Age   Private Subnet(s)
-------------------------------------------------------------------------------
stoweA            Current  4E89368B    680   169.254.70.0/24

  Tunnel          Peer            State      RTT(ms) Loss(%) TCP(Kb/s) Loss*RTT
  ---------------------------------------------------------------------------
  toBoston        bstnA           Connected      0.1     0.6   122070        1

prtlndA(cfg)# ...
```

Showing the RON Routes

Each ARX has a private IP subnet over which its internal processors communicate. When two or more switches are joined in a RON, the processors on one switch send packets through a RON tunnel to the processors on another switch; this requires an IP route from one private subnet to the other. The show ron route command shows the routing table for the current switch:

show ron route

This shows one row for each tunnel. In addition to the private-subnet destination and tunnel name, the average Round-Trip Time is shown under the Milliseconds heading.

For example:

```
prtlndA(cfg)# show ron route

Default Policy
--------------

  Destination            Subnet           via Tunnel        Milliseconds
  ------------------------------------------------------------------------
  bstnA               169.254.80.0/24     toBoston               0.1
  canbyA             169.254.116.0/24     toBoston               0.2
  newptA             169.254.52.128/26    toBoston               0.2
  provA               169.254.3.64/26     toBoston               0.2
  prtlndB             169.254.58.0/24     Redundancy               0
  stkbrgA              169.254.4.0/24     toBoston               0.3
  stoweA              169.254.70.0/24     toBoston               0.2
prtlndA(cfg)# ...
```

Removing a RON Tunnel

Use no ron tunnel to remove a tunnel.

no ron tunnel

If the tunnel is connected, the CLI warns you before removing it; enter **yes** to continue.

For example, this command sequence removes a tunnel from VLAN 25's management interface:

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# no ron tunnel toStageSys
Tunnel ''toStageSys'' is currently connected.
Delete tunnel ''toStageSys''? [yes/no] yes
bstnA(cfg-if-vlan[25])# ...
```

Starting a Remote CLI Session

You can use the rconsole command to traverse a RON tunnel and start a new CLI session on a remote ARX. Run this command from priv-exec mode. The remote CLI access occurs through a Secure Shell (SSH) session.

rconsole *hostname* [*username*]

where

hostname (1-128 characters) is the hostname for the remote switch. The remote switch must be reachable using a RON tunnel, and the name of the switch must be known to RON. To see the current switch names available through RON, use the show ron command (described above). Those switches showing the connection status 'Connected' are available through the rconsole command.

username (optional, 1-32 characters) is a valid administrative username on the remote ARX. (For information on administrative accounts, recall Appendix 2, Managing Administrative Accounts.) If you omit this, it defaults to your current administrative account, which may not exist on the remote ARX; only redundant switches are guaranteed to share the same user accounts and passwords.

The remote switch challenges you for your password. Use a password that works for the ***username*** on the remote ARX.

For example, the following command sequence exits to priv-exec mode and starts a CLI session on the switch, 'prtlndA,' using 'admin' as a username:

```
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# end
bstnA# rconsole prtlndA admin
Password: myP@55w0RD
prtlndA>
```

Setting a Default RON User

You have the option to create a default RON user for these commands. This is a valid administrative account on a remote ARX, to be used for the rconsole command whenever you omit a ***username***. Without this, the default is the current administrative account, which may not exist at the remote ARX.

From cfg mode, use ip ron-user to set the default username for logins over the RON:

```
ip ron-user username
```

where ***username*** is 1-32 characters.

The CLI then prompts twice for the password.

For example, the following command sequence sets up a default RON user named "juser:"

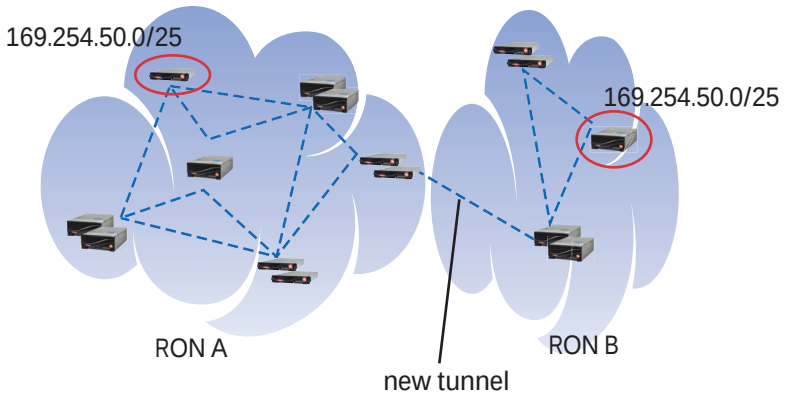
```
bstnA(cfg)# ip ron-user juser
Password: jpasswd
Validate Password: jpasswd
bstnA(cfg)# ...
```

Resolving Conflicting Subnets

RON communication breaks down when two switches in the RON have the same private subnet. Each processor in the switch has one or more home addresses on this subnet; if multiple switches have the same home addresses, communication to both switches is impossible. A chassis is assigned a subnet at the factory, from a range of up to 1024 possible subnets, so this situation is very rare in small RONs.

Whenever a tunnel is formed, the switch checks for conflicts amongst all private subnets at both ends of the tunnel. (Each end of the tunnel could already be part of a smaller RON, as illustrated below.) If a conflict exists, the conflicting switches are quarantined from any peers that are *new* to them. If the conflicting switches were previously part of a smaller RON, they continue to communicate with previously-connected switches.

For example, suppose two switches conflict as shown below. All switches in RON A continue to communicate to their local switch and ignore the conflicting switch in RON B. The switches in RON B ignore the conflicting switch in RON A. All non-conflicting switches can communicate to all other non-conflicting switches.



If a switch has a subnet conflict, its Status in the show ron command is "SUBNET CONFLICT." For example, the following command shows that a switch named "ARX-500-SJC" has a private-subnet conflict:

```
prtlndA(cfg)# show ron

Switch Name             HA Peer Switch                                Uptime
Status                  UUID                                 Management Addr
-------------------------------------------------------------------------------
bstnA                   (None)                              0 days, 02:07:57
ONLINE                  d9bdece8-9866-11d8-91e3-f48e42637d58          10.1.1.7

provA                   (None)                              0 days, 02:08:11
ONLINE                  db922942-876f-11d8-9110-8dtu78fc8329        10.1.38.19

prtlndA                 (None)                              0 days, 02:07:59
ONLINE                  876616f6-79ac-11d8-946f-958fcb4e6e35        10.1.23.11

ARX-500-SJC             (None)                             14 days, 08:00:20
SUBNET CONFLICT         e4556df2-45e8-11d8-9110-8d45e8fce455      172.16.77.60
```

```
...

prtlndA(cfg)#
```

Viewing All Subnet Conflicts

To view all conflicting switches in the RON, use the show ron conflicts command:

```
show ron conflicts
```

Each row shows one switch with a subnet conflict, along with the switch that conflicts with it. For example, the following command shows that "ARX-500-SJC" has a conflict with "BIG4000-WEST:"

```
prtlndA(cfg)# show ron conflicts

Accessible Switch        Conflicting Switch
-------------------------------------------------
BIG4000-WEST             ARX-500-SJC

prtlndA(cfg)#
```

Reassigning a Private Subnet

You must change the private subnet of any and all conflicting switches to resolve the conflict. This causes the switch to reboot and then join the RON normally. From cfg mode, use the ip private subnet reassign command to reassign the private subnet on the current chassis:

```
ip private subnet reassign
```

The CLI prompts for confirmation before changing the private subnet and rebooting the switch. Answer **yes** to proceed.

For example, this command sequence reassigns the subnet at "ARX-500-SJC:"

```
ARX-500-SJC(cfg)# ip private subnet reassign
Reassign a new, unused, private subnet and reboot the chassis? [yes/no] yes
...
```

Evicting a Defunct Switch from the RON

After a switch is removed from the RON, the show ron command continues to display the defunct switch. This facilitates switch replacement, where the installer copies the UUID of the defunct chassis onto the new one (see the appropriate *Hardware Installation Guide* for full instructions). There are some situations where no replacement is planned: for these cases, you can remove the defunct switch information from the show ron output. From priv-exec mode, you can use the ron evict command to remove all traces of the switch:

ron evict *hostname*

where ***hostname*** identifies the switch to evict (for example, “demo-a1k”).

The switch must be offline and/or disconnected from the RON (that is, no working tunnels) for this command to succeed.

For example, the following command sequence shows an offline switch, exits to priv-exec mode, and evicts it from the RON:

```
bstnA(cfg)# show ron expir626-1k


  Switch Name:           expir626-1k
  HA Peer Switch:        (None)
  Status:                OFFLINE
  Uptime:                0 days, 14:34:49
  UUID:                  9a6eb9ac-6c6d-11d8-9444-9e00f495ff7e

bstnA(cfg)# end
bstnA# ron evict expir626-1k
bstnA# ...
```

7

Adding a Redundant Switch

Overview

Use this chapter to join two ARXes together as a redundant pair, also known as a High-Availability (HA) pair. Redundant switches monitor one another for failure: if one switch fails, all of its services fail over to its peer.

Note

The ARX-VE is a software-only virtual appliance, and, as such, functionality related to physical network infrastructure is not relevant to its operation. Redundant, high-availability support for the ARX-VE is available via standard hypervisor clustering functionality.

Before You Begin

The peers in a redundant pair must have matching hardware configurations. An ARX-2000 can only be paired with another ARX-2000, an ARX-4000 can only be paired with another ARX-4000, and so on.

Switch Installation

When installing the redundant peer, you must use the same *master key* as the original switch. The master key is an encryption key for all switch passwords: the switches share all the same administrator accounts and groups, so they require the same master key. You enter the master key in the initial startup script for the switch.

Nearly all other initial-startup parameters are the same for both peers: the only exceptions are the out-of-band-management IP address and the private subnet. Each peer has a unique management IP so that you can separately manage the switches. Private messages are exchanged between the peers' private subnets, so those subnets must be distinct.

The installation manuals have complete instructions for installing a redundant switch, including these details. Follow those installation instructions before configuring redundancy between the switches:

- ARX-500 - *ARX®–500 Hardware Installation Guide*
- ARX-1500 - *ARX®–1500 Hardware Installation Guide*
- ARX-2000 - *ARX®–2000 Hardware Installation Guide*
- ARX-2500 - *ARX®–2500 Hardware Installation Guide*
- ARX-4000 - *ARX®–4000 Hardware Installation Guide*

Concepts and Terminology

Every pair has a *senior* switch and a *junior* switch. The seniority of each switch changes with each failover. At initial redundancy configuration, seniority is determined based on which switch has namespaces and/or global servers:

- If neither switch has any namespaces or global servers, the switch with the lower rendezvous address (explained below) is senior.
- If only one switch has them, that switch is senior.
- If both switches have them, the pair cannot form. This chapter explains how to detect and recover from this problem.

If the senior switch fails, all services fail over to the junior switch, which then becomes the senior switch.

The *role* of each peer indicates whether or not it is capable of running any client services. A peer's role is either *active* or *backup*. The senior switch is active and the junior switch is backup. You cannot change any global configuration (such as namespaces and global servers) from a backup switch.

Namespaces and Global Servers

A *namespace* is a collection of file systems under a single authentication domain. The *ARX® CLI Storage-Management Guide* explains how to aggregate the storage of multiple external filers into a single namespace, where each filer share (or export) is virtualized as one *namespace share*. Clients can access a namespace's aggregated storage through a front-end service, such as CIFS or NFS, running on a global server.

A *global server* contains front-end services for the clients to use, such as CIFS and NFS. Clients access a global server through its *virtual server*, which contains a virtual-IP (*VIP*) address. The virtual server and its VIP fail over from peer to peer, so the global server and its services are highly available. Global and Virtual-Server configuration is also described in the *ARX® CLI Storage-Management Guide*.

Namespaces and global servers are the most-important components of the *global config*. The global config is similar to the running config, but it is shared between redundant peers.

Synchronizing the Network Configurations

The first step in configuring redundancy is to synchronize the L2 and L3 settings on both switches. Both peers require the same access to the client and server networks. However, you configure unique proxy-IP addresses for each of them, so that they can connect to back-end servers simultaneously. These parameters are discussed below, followed by a sample configuration for both peers.

Using the Same Subnets for VIPs and Proxy IPs

The peers must share the same visibility to clients and servers in order to support the same services. All clients must be on the same subnet as their global/virtual server, or the global/virtual server must be able to reach its clients through a static route. All servers must reside on (or be reachable through) the subnet with all the proxy-IP addresses. You must configure both peers so that they have equivalent visibility to the network, possibly by using static routes (see *Adding a Static Route*, on page 4-10).

Using Unique IP Addresses

The peers' network configurations have different proxy IPs and management addresses. These are "home" addresses for each switch; they must be different so that both switches can simultaneously contact the back-end filers, and so that you can manage them separately. After you configure these IP addresses at one peer, configure a different set of addresses at the other. Refer back to the following sections for configuration details:

- *Configuring an In-Band (VLAN) Management Interface*, on page 4-4,
- *Adding a Range of Proxy-IP Addresses*, on page 4-7, and
- *Changing the Out-of-Band-Management Interface (optional)*, on page 4-18. The out-of-band management interface is set during the initial startup script described in the hardware-installation manual.

The proxy-IP addresses, while different, must reside on the same subnet and VLAN.

Using a Unique Hostname

Each switch must have a unique hostname to allow them to form a redundant pair. The hostname appears in the CLI prompt. From cfg mode, use the hostname command to change it.

Checking for Private Subnet Conflicts

A private subnet is assigned to each chassis at the factory, from a pool of 256-1024 subnets (depending on chassis type). Both peers must have different private subnets to function as a redundant pair. You can use the show ron database command (recall *Showing the RON Database*, on page 6-8) on both switches to verify that their private subnets are different. If the private subnets match, use the ip private vlan ... subnet command to assign a new subnet to one of the chassis' (this reboots the chassis; see *Changing the Private Subnet*, on page 3-12).

Sample Configuration

For example, the following command sequence configures network settings for two switches, Peer A ("prtlndA") and Peer B ("prtlndB"). (The out-of-band management interfaces and private subnets were set up as part of their respective startup scripts.)

Presumably, Peer A was configured before Peer B was first booted; Peer A's configuration is shown below for comparison purposes. Differences between the configurations are noted in the comments:

Peer A

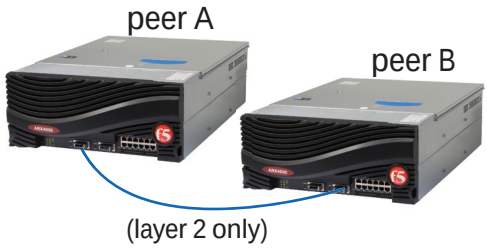
```
; Enter cfg mode and set the hostname.
SWITCH# config
SWITCH(cfg)# hostname prtlndA
; Assign a range of proxy IPs from 192.168.74.0/24.
prtlndA(cfg)# ip proxy-address 192.168.74.21 255.255.255.0 vlan 74 count 4
; A default route through the 192.168.74.1 gateway, plus
; one static route to a second client subnet at 192.168.78.0/24:
prtlndA(cfg)# ip route 0.0.0.0 0.0.0.0 192.168.74.1 255
prtlndA(cfg)# ip route 192.168.78.0 255.255.255.0 192.168.74.2
```

Peer B

```
; Enter cfg mode and set the hostname.
SWITCH# config
SWITCH(cfg)# hostname prtlndB
; This switch uses a different range of proxy IPs
; from the same subnet as Peer A.
prtlndB(cfg)# ip proxy-address 192.168.74.41 255.255.255.0 vlan 74 count 4
; Same default/static routes:
prtlndB(cfg)# ip route 0.0.0.0 0.0.0.0 192.168.74.1 255
prtlndB(cfg)# ip route 192.168.78.0 255.255.255.0 192.168.74.2
```

Preparing Interfaces for the Redundancy Link

The next step in redundancy configuration is to set up layer 2 for the redundant pair's link. The redundant pair uses this link to exchange heartbeats and synchronization data. Cable the switches together (a direct, gigabit or ten-gigabit connection is strongly recommended) and use the cfg-if-gig or cfg-if-ten-gig redundancy protocol command to prepare the connected interfaces for redundant-link communication: see *Preparing for Use in a Redundant-Pair Link*, on page 3-53.



Note

These instructions do not apply to ARX-1500 and ARX-2500, for which the redundant-pair link is a VLAN. In this case, an interface can be associated with a VLAN, and that VLAN prepared in turn for use in a redundant-pair link, as described in Preparing for Use in a Redundant-Pair Link, on page 3-53.

Note

The ARX-500 has a dedicated port for its redundancy link, 1/2, so these commands are unnecessary. If you are joining two ARX-500 chassis, cable the two ports together to establish a layer-2 connection between the peers, then skip ahead to the next section (Creating a Tunnel to the Peer (ARX-500 Only), on page 7-11).

For example, the following command sequence prepares port 2/1 for use as a redundant-link interface:

```
bstnA(cfg)# interface ten-gigabit 2/1
bstnA(cfg-if-ten-gig[2/1])# redundancy protocol
bstnA(cfg-if-ten-gig[2/1])# exit
bstnA(cfg)# ...
```

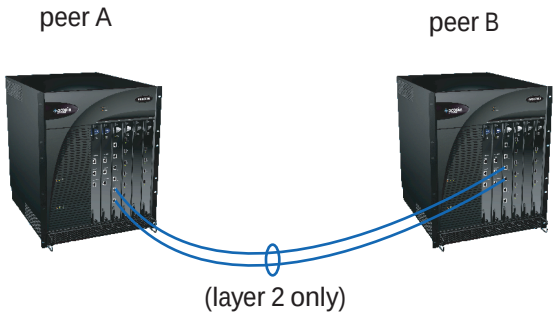
Connection Speed and Latency

A Gigabit connection with low latency is required for optimal failover performance. 100MB connections are supported, but cause an interruption in service when a failed switch comes back online: synchronizing data between the peers can take several minutes over the slower connection.

F5 Networks recommends a direct connection between redundant peers. It is a best practice to have the peers in the same location. A connection through a Gigabit L2 switch is permissible, provided the speed is high and the latency is low.

Preparing a Channel

F5 Networks recommends strongly that you configure a link-aggregation channel for the redundancy link to ensure resilient and optimized performance. If one port fails, the other(s) continue to carry traffic. The cfg-channel version of redundancy protocol prepares the channel for redundancy and adds interfaces to it at the same time. Refer to *Preparing a Channel as a Redundant-Pair Link*, on page 3-37.



Note

These instructions do not apply to ARX-1500 and ARX-2500, for which the redundant-pair link is a VLAN. In this case, a channel can be associated with a VLAN, and that VLAN prepared in turn for use in a redundant-pair link, as described in Preparing for Use in a Redundant-Pair Link, on page 3-53.

For example, the following command sequence takes place on two ARX-2000 peers with a channel connection. For Peer A, “prtlndA,” the command sequence prepares two ports for use in a channel, then configures a redundancy-ready channel with those ports. The process is then repeated at Peer B, “prtlndB:”

Peer A

```
prtlndA(cfg)# interface gigabit 1/11
prtlndA(cfg-if-gig[1/11])# speed 1000-full
prtlndA(cfg-if-gig[1/11])# no shutdown
prtlndA(cfg-if-gig[1/11])# exit
prtlndA(cfg)# interface gigabit 1/12
prtlndA(cfg-if-gig[1/12])# speed 1000-full
prtlndA(cfg-if-gig[1/12])# no shutdown
prtlndA(cfg-if-gig[1/12])# exit
prtlndA(cfg)# channel 1
prtlndA(cfg-channel[1])# redundancy protocol 1/11 to 1/12
prtlndA(cfg-channel[1])# lacp passive
prtlndA(cfg-channel[1])# exit
prtlndA(cfg)# ...
```

Peer B

```
prtlndB(cfg)# interface gigabit 1/11
prtlndB(cfg-if-gig[1/11])# speed 1000-full
prtlndB(cfg-if-gig[1/11])# no shutdown
prtlndB(cfg-if-gig[1/11])# exit
prtlndB(cfg)# interface gigabit 1/12
prtlndB(cfg-if-gig[1/12])# speed 1000-full
prtlndB(cfg-if-gig[1/12])# no shutdown
prtlndB(cfg-if-gig[1/12])# exit
prtlndB(cfg)# channel 1
prtlndB(cfg-channel[1])# redundancy protocol 1/11 to 1/12
prtlndB(cfg-channel[1])# lacp passive
prtlndB(cfg-channel[1])# exit
prtlndB(cfg)# ...
```

Support For Stretch Clusters

Stretch clusters are redundant ARX pairs in which the two switches in the cluster are separated geographically, but in which the redundancy interface is a direct connection between the two switches, without an intervening switch (though, very likely, through a patch panel).

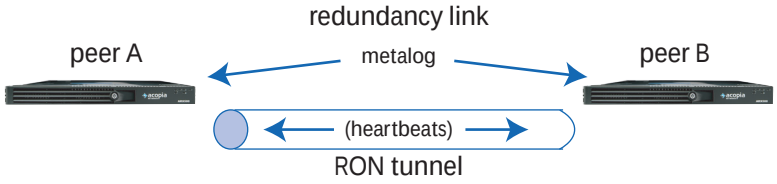
Stretch cluster functionality is supported explicitly for the ARX-2500. Long-reach optical connectors and short-reach copper connectors are available for supporting low-latency direct connections over the ARX-2500's 10-Gigabit Ethernet interfaces.

Several CLI commands are particularly relevant when using stretch clusters; refer to the CLI Reference Guide for a detailed description of each:

- resilver-timeout: The redundant peers exchange their metalog data during initial rendezvous, or after a failover. This is called resilvering the metalog data. At some sites, the latency between peers is high enough that the resilvering process times out before the pair can form; this command enables you to control that timeout interval.
- show redundancy resilver-timeout: This displays the value of the resilver timeout.
- show redundancy metalog: This displays metalog performance statistics for the redundant pair. This consists of two different sets of statistics, depending upon the HA pair's operational state. While resilvering is occurring, it provides incremental progress of resilvering as it is being performed, as well as an estimated amount of time to completion. When resilvering is complete and the switches have formed an HA pair, the command displays statistics for periodic metalog updates (termed "steady state").

Creating a Tunnel to the Peer (ARX-500 Only)

The ARX-500 has a dedicated port for its redundancy link. This redundancy link carries synchronization data between the peers. It does not carry redundancy heartbeats (discussed later in the chapter); you must configure an explicit RON tunnel for heartbeat traffic. This only applies to the ARX-500 platform; skip ahead to the next section if you are pairing any other chassis types.



As explained in the previous chapter, a RON tunnel terminates at an inband (VLAN) management interface. From cfg-if-vlan mode, use the ron tunnel command to create one end of a RON tunnel. (For details, refer back to *Creating a Tunnel to Another ARX*, on page 6-4.) This puts you into cfg-if-vlan-ron-tnl mode, where you must identify the IP address at the other end of the tunnel and enable the local end.

Repeat the RON-tunnel configuration on both peers.

For example, the following command sequence configures a RON tunnel between two ARX-500 switches, provA and provB:

Peer A

```
provA(cfg)# interface vlan 103
provA(cfg-if-vlan[103])# ip address 192.168.103.160 255.255.255.0
provA(cfg-if-vlan[103])# no shutdown
provA(cfg-if-vlan[103])# ron tunnel toRedPeerB
provA(cfg-if-vlan-ron-tnl[103~toRedPeerB])# peer address 192.168.103.161
provA(cfg-if-vlan-ron-tnl[103~toRedPeerB])# no shutdown
provA(cfg-if-vlan-ron-tnl[103~toRedPeerB])# ...
```

Peer B

```
provB(cfg)# interface vlan 103
provB(cfg-if-vlan[103])# ip address 192.168.103.161 255.255.255.0
provB(cfg-if-vlan[103])# no shutdown
provB(cfg-if-vlan[103])# ron tunnel toRedPeerA
provB(cfg-if-vlan-ron-tnl[103~toRedPeerA])# peer address 192.168.103.160
provB(cfg-if-vlan-ron-tnl[103~toRedPeerA])# no shutdown
provB(cfg-if-vlan-ron-tnl[103~toRedPeerA])# ...
```

Configuring Redundancy For ARX-1500 and ARX-2500

This section summarizes the processes of configuring networking behavior and HA behavior for the ARX-1500 and the ARX-2500.

This section covers the following information on configuring redundancy for these platforms:

- *Configuration Summary* lists the essential actions that you must execute to configure an ARX-1500 pair or ARX-2500 pair for redundancy.
- *Configuration Example* provides examples of the CLI commands that you must execute to configure an ARX-1500 pair or ARX-2500 pair for redundancy.

Although these configuration processes are largely the same as they are for other ARX platforms, changes in the designs of these newer ARX platforms require that some aspects of their networking and HA configurations differ from the configuration procedures described for the other ARX platforms. For complete details of commands and configuration steps that are common across all ARX platforms, please refer to each section that is referenced, in which the configuration is described in detail. Refer also to the *ARX® CLI Reference* for a comprehensive description of each CLI command and its associated parameters.

Note the following differences in the ARX-1500 and ARX-2500 platforms:

- The ARX-1500 and ARX-2500 do not support layer 2 switching.
- The ARX-1500 and ARX-2500 do not provide dedicated management interfaces. Instead, port 1/1 is configured by default as the out-of-band management interface.
- The ARX-1500 and ARX-2500 have only the default VLAN (VLAN 1) configured by default. They do not have either a private VLAN (VLAN 1002 on other ARX platforms) or a metalog VLAN (VLAN 1003 on other ARX platforms) configured in their factory default configuration.

As a result of these platform differences, the ARX-1500 and ARX-2500 must be configured differently for redundancy:

- A separate link (separate from the management interface) must be configured to carry heartbeat and metalog data.
- The separate link must be on a different IP subnet than the management interface.
- The separate link is configured with a VLAN.

Configuration Summary

This section lists the configuration tasks that you must perform to configure an ARX-1500 or ARX-2500 pair for HA redundancy. Although there are numerous configuration actions that are supplementary or optional, those listed here are essential.

Execute the following actions for each chassis in the pair:

- Configure the layer 2 interface that will be used for the redundancy link. This can be a single port or a single channel (recommended), and must be separate from the management interface used to access the chassis initially. Follow the instructions in *Configuring a Layer-2 Interface*, on page 3-23 if you are configuring a single port, or follow the instructions in *Configuring a Channel (802.3ad Link Aggregation)*, on page 3-35 if you are configuring a channel.
- Add a VLAN for use as the redundancy link, using the vlan CLI command as described in *Adding a VLAN*, on page 3-6. This VLAN must be configured explicitly, because the ARX-1500 and the ARX-2500 do not have a private or metalog VLAN configured in their factory default configurations.
- Add to the VLAN the port or channel that you configured previously. As appropriate, follow the instructions in *Adding Ports to the VLAN*, on page 3-6 or *Adding the Channel to a VLAN*, on page 3-35. Note that for ARX-1500 and ARX-2500, a channel must be added to a VLAN explicitly; channels on these platforms are not added to any VLAN by default.
- Prepare the VLAN for redundancy, using the redundancy CLI command in cfg-interface-vlan mode as described in *Preparing for Use in a Redundant-Pair Link*, on page 3-53, in the subsection *ARX-1500 and ARX-2500*, on page 3-54.
- Configure the VLAN's management IP address, as described in *Setting the Management IP Address*, on page 4-4.

Configuration Example

This section shows an example configuration. Substitute IP addresses and resource names as appropriate to reflect your own configuration.

Perform this sequence of actions first on one member of the redundant pair, then on the other:

Configure the layer 2 interface that will be used in the redundancy link; this can be a single port or a single channel (recommended). Do not include the existing management interface.

Channel:

```
canbyA(cfg)# channel 1
canbyA(cfg-channel[1])# members 1/7 to 1/8
canbyA(cfg-channel[1])# exit
canbyA(cfg)#
```

Add the VLAN to be used as the redundancy link:

```
canbyA(cfg)# vlan 122
canbyA(cfg-vlan[122])#
```

Add the port or channel to the VLAN:

Port:

```
canbyA(cfg-vlan[122])# members 1/2
canbyA(cfg-vlan[122])# exit
```

```
canbyA(cfg)#
```

Channel:

```
canbyA(cfg)# channel 1
canbyA(cfg-channel[1])# vlan 122
canbyA(cfg-channel[1])# exit
canbyA(cfg)#
```

Assign an IP address to the VLAN, using a different subnet than that used for the management interface:

```
canbyA(cfg-if-vlan[122])# ip address 192.168.122.9
255.255.255.0
canbyA(cfg-if-vlan[122])#
```

Configure the VLAN to support redundancy:

```
canbyA(cfg-if-vlan[122])# redundancy
canbyA(cfg-if-vlan[122])# exit
canbyA(cfg)#
```

Specify the chassis' redundancy peer:

```
canbyA(cfg)# redundancy
canbyA(cfg-redundancy)# peer 192.168.122.8
```

Repeat this sequence of actions on the other member of the redundant pair.

Configuring Redundancy

The next step is to set up the redundancy parameters on each peer. From cfg mode, use the redundancy command to begin this process:

redundancy

This puts you into cfg-redundancy mode, where you identify the peer's IP address, set up a shared *quorum disk* for the redundant pair (explained below), and enable redundancy processing.

For example:

```
prtlndA(cfg)# redundancy
prtlndA(cfg-redundancy)# ...
```

Identifying the Peer Switch

The first redundancy parameters are the IP addresses for the initial rendezvous. You use one of the peer's management IPs as this address. This can be the out-of-band-management interface (labeled MGMT on the front panel; recall *Changing the Out-of-Band-Management Interface (optional)*, on page 4-18) or one of the inband-management interfaces associated with a VLAN; if the latter, the interface must be pre-configured for redundancy usage (see *Designating for Redundancy (optional)*, on page 4-5).

The interface address is used for the initial *rendezvous* between the peer switches, when they exchange configuration data and enable the redundancy heartbeats. You start the rendezvous later, when you enable redundancy.

From cfg-redundancy mode, use the peer command to identify the peer:

peer *management-ip*

where ***management-ip*** is a valid management-IP address at the peer (for example, 192.168.9.12).

For example, the following command sequence uses the out-of-band management IPs for each of the ARX-2000 peers:

Peer A

```
prtlndA(cfg)# redundancy
prtlndA(cfg-redundancy)# peer 10.1.23.12
prtlndA(cfg-redundancy)# ...
```

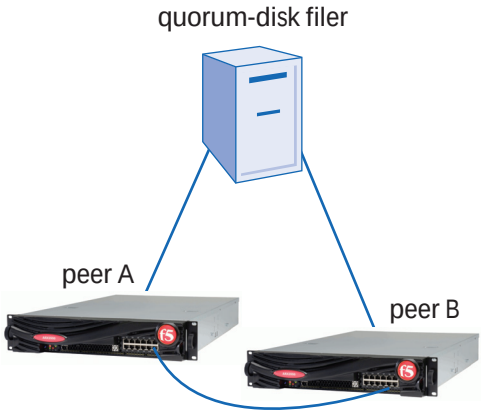
Peer B

```
prtlndB(cfg)# redundancy
prtlndB(cfg-redundancy)# peer 10.1.23.11
prtlndB(cfg-redundancy)# ...
```

Adding the Quorum Disk

The next step in redundancy configuration is choosing a quorum disk for the redundant pair. A *quorum disk* is an external share that provides an additional path for heartbeats. The peers write their heartbeat messages onto the quorum disk in addition to sending them over the redundant-pair link.

This provides redundant paths for redundancy communication. The peers use heartbeat status to help determine each other's up/down status: if only one heartbeat fails, the switch knows that its peer is still up.



The quorum disk also records ballot information for senior-switch election.

You must set identical quorum-disk configurations at both switches for the redundant pair to form.

Requirements for the Filer and Share

Use a highly-available, high-performance filer for the quorum disk. The round-trip time for writing 1 block of quorum-disk data must be less than 1 second. Slow-performing filers may cause the redundancy state to fluctuate and could lead to unnecessary down time.

The quorum disk can be any writable external-filer share. It *cannot* be imported as part of a managed volume; however, the share can reside on a filer that also hosts volume shares. The share must have a minimum of 1 MB of free space. You can use an NFS or CIFS share, as explained below.

Reaching the Filer Through an In-Band-Management IP

The ARX uses an in-band (VLAN) management interface to communicate with the quorum disk. For a switch with multiple in-band-management interfaces, it chooses the one with the best path to the filer. Use the interface vlan command to create an in-band management address for a particular VLAN, as shown in *Configuring an In-Band (VLAN) Management Interface*, on page 4-4. The switch can reach the quorum-disk filer through this interface only if

- the filer is on the same VLAN as the management interface, or
- a static route through the interface's VLAN can reach the filer. (That is, the route's gateway must be on the same VLAN as the in-band-management interface.) For instructions on configuring a static route, refer back to *Adding a Static Route*, on page 4-10.

Using an NFS Share

From cfg-redundancy mode, use the quorum-disk command to identify the quorum disk's filer, share, and protocol. This is the syntax for an NFS quorum disk:

quorum-disk *ipaddress*:*/path* {nfs2 | nfs3 | nfs3tcp}

where

ipaddress*:*/path can be up to 1024 characters long:

- ***ipaddress*** must be on the proxy-IP subnet (see *Adding a Range of Proxy-IP Addresses*, on page 4-7) or reachable through a static route (see *Adding a Static Route*, on page 4-10), and
- ***path*** identifies the NFS share.

Note

An NFS share must be configured for synchronous writes. Use the "sync" option in /etc/exports; for most implementations of NFS, this is the default. As added failure protection, we recommend you also turn off write delay (using the "no_wdelay" option).

For example, the following line from /etc/exports is valid for a quorum-disk share:

/var/quorum *(rw,no_root_squash,sync,no_wdelay)

nfs2 | nfs3 | nfs3tcp is a required choice. The **nfs3** option (without the **tcp** at the end) denotes NFSv3 over UDP.

The following sample-command sequence uses an NFS share as a quorum disk. The same settings are used at both peers:

Peer A

```
prtlndA(cfg)# redundancy
prtlndA(cfg-redundancy)# quorum-disk 192.168.74.83:/lhome/quorum-disk/portland1 nfs2
prtlndA(cfg-redundancy)# ...
```

Peer B

```
prtlndB(cfg)# redundancy
prtlndB(cfg-redundancy)# quorum-disk 192.168.74.83:/lhome/quorum-disk/portland1 nfs2
prtlndB(cfg-redundancy)# ...
```

Using a CIFS Share

You can use the same command to choose a CIFS share. Substitute a CIFS-share-path format for the NFS-share format, and add a windows username at the end:

quorum-disk *ipaddressshare*[*path*] cifs [*DOMAIN/*]*username* [spn *spn*]**

where

***ipaddress**share*[*path*]** can be up to 1024 characters long.

- ***ipaddress*** must be on the proxy-IP subnet (see *Adding a Range of Proxy-IP Addresses*, on page 4-7) or reachable through a static route (see *Adding a Static Route*, on page 4-10), and
- ***share*[*path*]** identifies the CIFS share, and possibly a path within the share. This share should, if possible, block on any write request until the data has been committed to disk.

cifs is a required keyword.

[*DOMAIN/*]*username* can also be up to 1024 characters:

- ***DOMAIN*** (optional) is the user's domain name (for example, BOSTONCIFS/), and
- ***username*** is the username that the redundancy software uses to read and write heartbeat messages on the quorum disk. Choose a username with read/write permissions at the share.

spn *spn* (1-256) is optional in general, but required for a Windows 2008 cluster. This is the Service Principal Name (SPN) for the CIFS service with the above ***share***. A name is required to connect to a CIFS service from a Windows 2008 Cluster; the IP address is not sufficient.

The CLI prompts for a password, and then prompts to confirm it.

For example, the following command sequence uses a CIFS share as a quorum disk. The username and password are jqpublic/jqpasswd, the domain is MEDARCH.ORG, and the SPN is cifssrv@MEDARCH.ORG. As with the NFS quorum disk, the configurations must match on both switches.

Peer A

```
prtlndA(cfg)# redundancy
prtlndA(cfg-redundancy)# quorum-disk \\192.168.74.214\q_disk cifs MEDARCH.ORG/jqpublic spn
cifssrv@MEDARCH.ORG
Password: jqpasswd
Confirm: jqpasswd
prtlndA(cfg-redundancy)# ...
```

Peer B

```
prtlndB(cfg)# redundancy
prtlndB(cfg-redundancy)# quorum-disk \\192.168.74.214\q_disk cifs MEDARCH.ORG/jqpublic spn
cifssrv@MEDARCH.ORG
Password: jqpasswd
Confirm: jqpasswd
prtlndB(cfg-redundancy)# ...
```

Identifying a Critical Route

Some external subnets may be required for the ARX to offer full service to its clients. You can identify these subnets as *critical routes*, where a failover to the redundant peer may occur if the switch loses all routes to the subnet. (Failover may not occur if it could exacerbate a service problem; this is discussed below.) From cfg-redundancy mode, use the critical route command to create one critical route:

critical route *subnet mask*

where

subnet is an IP address that identifies the critical subnet (for example, 172.16.78.0), and

mask shows the subnet part of the IP address (for example, 255.255.255.0).

The CLI prompts for confirmation if the route is down, the peer is down, or there is some other issue. Enter **yes** if you want to proceed anyway.

You can repeat this command to enter multiple critical routes. Any ***subnet*** must be reachable by at least one static route; use the show ip route command to show all static routes, and use the ip route command to create a new one. (Refer back to *Adding a Static Route*, on page 4-10 for instructions on using both ip-route commands.) The ARX tests the route by sending an ARP request to the route's gateway every 20 seconds.

◆ Note

Critical routes do not have to be mirrored on both peers. Both peers may have different network configurations, so they may have different sets of critical routes.

If you want to have the same critical routes on both peers, you must repeat these steps at the peer switch.

For example, the following command sequence establishes the default route as a critical route:

```
prtlndA(cfg)# redundancy
prtlndA(cfg-redundancy)# critical route 0.0.0.0 0.0.0.0
prtlndA(cfg-redundancy)# ...
```

If a critical route fails on the current peer and the other peer has no failures, control fails over to the other peer. If the other peer has any failures that would ordinarily cause a failover (such as a major hardware fault), no failover occurs. This prevents unnecessary failovers.

The ARX tests for failure with regular ARP requests. Every 20 seconds, the ARX sends an ARP to the route's gateway. (The gateway is configured with the ip route command.) If the gateway fails to respond, the ARX waits an additional 20 seconds before asking the peer if it is possible to fail over. The ARPs continue indefinitely at 20-second intervals. If the gateway responds before the failover is initiated, the failover does not occur.

The ARX uses an in-band (VLAN) management addresses to send those ARP requests, and the subnet you identify with the critical route command must have a management address for its VLAN. Use the interface vlan command to create the in-band management address for the subnet's VLAN.

Loss of a Critical Route Does Not Always Trigger Failover

A failover does not necessarily occur if an active switch loses connectivity to a critical route. If the redundant peer also has a redundancy problem (for example, it cannot reach one of its own critical routes), a failover does not occur. When both peers have failures, a failover would only decrease service availability.

Removing a Critical Route

Use the no critical route command to remove a critical route:

no critical route *subnet mask*

For example, the following command sequence removes the critical route to 172.16.99.0/24:

```
prtlndA(cfg)# redundancy
prtlndA(cfg-redundancy)# no critical route 172.16.99.0 255.255.255.0
prtlndA(cfg-redundancy)# ...
```

Enabling Redundancy

The final step in configuring redundancy is enabling it on both switches. From cfg-redundancy mode on either switch, use the enable command to enable redundancy:

enable

At most, only one of the peers should have any namespaces or global servers configured (these are described in the *ARX® CLI Storage-Management Guide*). If both peers have these objects defined, you must clear the entire global config on one of them before you can proceed (as described below).

For example, the following command sequence enables redundancy on "prtlndA," then enables it on its redundant peer, "prtlndB:"

Peer A

```
prtlndA(cfg)# redundancy
prtlndA(cfg-redundancy)# enable
prtlndA(cfg-redundancy)# ...
```

Peer B

```
prtlndB(cfg)# redundancy
prtlndB(cfg-redundancy)# enable
prtlndB(cfg-redundancy)# ...
```

You can use show redundancy history (described later in the chapter) to monitor the rendezvous process.

Stopping a Failed Rendezvous

If the pair cannot form for some reason, the switches continuously retry the rendezvous. You can stop the retries with the no enable command:

```
no enable
```

This does not work after the pair successfully forms.

For example, the following command sequence stops the “bstnB” switch from trying to connect to a redundant peer:

```
bstnB(cfg)# redundancy
bstnB(cfg-redundancy)# no enable
bstnB(cfg-redundancy)# ...
```

Showing the Redundancy Configuration

You can use the show redundancy command to show the high-level configuration parameters for the redundant pair, and/or to monitor a redundant-pair rendezvous. You can issue this command from either peer:

```
show redundancy
```

A functioning redundant pair shows three nodes (both peers and the quorum disk) with a status of "Up." Take note of the Role column: one switch is "Active" and the other assumes the "Backup" role. You can only enter gbl mode on the "Active" switch. A leading asterisk (*) indicates the local switch.

For example, this shows the redundant pair from the "prtlndA" switch:

```
prtlndA(cfg)# show redundancy

                                          Transitions
Node Switch/Quorum Disk   Status     Role    Total Last (UTC)
---- -------------------- ---------- ------- ----- -------------------
*1   prtlndA              Up         Active  Never -
 2   prtlndB              Up         Backup  1     05:33:19 09/14/2009
 QD  192.168.74.83        Up         Quorum  1     05:33:07 09/14/2009
prtlndA(cfg)# ...
```

Showing Details About the Peer

Use the show redundancy peer command to see the configuration and status for the redundant peer:

```
show redundancy peer
```

For example, this shows the "prtlndA" switch from "prtlndB," its redundant peer:

```
prtlndB# show redundancy peer

Peer
  Name:           prtlndA
  IP Address:     10.1.23.11
  Port:           49800

Status:           Active

Heartbeats
  Sent:           21
  Received:       20

Transitions
  Count:          1
  Last:           05:33:19 09/14/2009
  Reason:         Peer switch 'prtlndA' is now online
prtlndB# ...
```

Showing the Quorum Disk

Use the show redundancy quorum-disk command to see the configuration and status for the quorum disk, along with some latency statistics:

show redundancy quorum-disk

For example:

```
prtlndA# show redundancy quorum-disk

Path:             192.168.74.83:/exports/quorum-disk/portland1
Protocol:         nfs2
Status:           Up

Heartbeats
  Sent:           744
  Received:       741

Transitions
  Count:          1
  Last:           05:33:07 09/14/2009
  Reason:         Quorum disk 192.168.74.83:/exports/quorum-disk/portland1 is now online.

Heartbeat Latency:
                        Heartbeat Latency Intervals (msec)
  Time Interval    [0-499]   [500-999]   [1000-3999]   [No Response]
  -------------    ------------------------------------------------
  01:00 - 01:45       745           0              0               0
  00:00 - 01:00         0           0              0               0
  23:00 - 24:00         0           0              0               0
  22:00 - 23:00         0           0              0               0
  21:00 - 22:00         0           0              0               0
  20:00 - 21:00         0           0              0               0
  19:00 - 20:00         0           0              0               0
  18:00 - 19:00         0           0              0               0
  17:00 - 18:00         0           0              0               0
  16:00 - 17:00         0           0              0               0
  15:00 - 16:00         0           0              0               0
  14:00 - 15:00         0           0              0               0
  13:00 - 14:00         0           0              0               0
  12:00 - 13:00         0           0              0               0
  11:00 - 12:00         0           0              0               0
  10:00 - 11:00         0           0              0               0
  09:00 - 10:00         0           0              0               0
  08:00 - 09:00         0           0              0               0
  07:00 - 08:00         0           0              0               0
  06:00 - 07:00         0           0              0               0
  05:00 - 06:00         0           0              0               0
  04:00 - 05:00         0           0              0               0
  03:00 - 04:00         0           0              0               0
  02:00 - 03:00         0           0              0               0

Heartbeat latency summary:
  0-499 msec            : 100.00%
  500-999 msec          :   0.00%
  1000-3999 msec        :   0.00%
  No response           :   0.00%

prtlndA# ...
```

Quorum Disk Should Have Few Transitions and Low Latency

The Transitions -> Count field should have a very low number, the Transitions -> Last field should not typically show a recent date, and the heartbeat latency should almost always be less than 999 milliseconds (msecs). Quorum-disk transitions should only occur on the first rendezvous and when you perform a scheduled upgrade on the quorum-disk filer. Communication with this disk continues even during a failover. The latency should be continuously low, and the filer should be highly available.

A large transition count, a recent last-transition date, and/or high latency may indicate poor communication with the quorum disk and/or an unreliable filer. Check for this periodically and correct it if you see a problem. You can correct it by choosing a better quorum disk: do this by changing the quorum disk at both peers (recall *Adding the Quorum Disk*, on page 7-15).

Showing the Critical Services

Critical services are critical routes, the quorum disk, and critical namespace shares.

- critical routes - if the senior switch loses contact with a critical route, a failover may occur (as explained above).
- quorum disk - the quorum disk is considered a *latent* critical service.
  - If an active switch loses contact with the quorum disk, it does *not* initiate a failover.
  - If an active switch loses some other critical service, it initiates a failover. If the peer switch has all of its critical services, a failover occurs.
  - The critical nature of the quorum disk appears in the final case: if an active switch has reason to fail over but the peer switch has lost contact with the quorum disk, *no failover occurs*.
- namespace share - if an active switch loses contact with a share that it needs to keep providing service, it initiates a failover. As with critical routes, a failover occurs only if the peer switch has all of its critical services intact.

◆ (Namespaces are described in the *ARX® CLI Storage-Management Guide*, along with the CLI command to designate a namespace share as critical.)

Use the show redundancy critical-services command to see all critical services on the current peer:

```
show redundancy critical-services
```

For example, the “prtlndA” switch has two critical metadata shares, the quorum disk, one critical route, and one critical share on its list of critical services:

```
prtlndA# show redundancy critical-services
```

```
                                                    Transitions
Type        Service                           Status  Count Last (UTC)
-------------------------------------------------------------------------------
meta-only   192.168.74.89:/vol/vol1/mdata_A     Up       1  2007-02-01 18:22:34
meta-only   192.168.74.89:/vol/vol1/mdata_B     Up       1  2007-02-01 18:22:54
quorum      192.168.74.83:/lhome/quorum-disk/portland1 Up       1  2007-02-01 18:20:14
route       0.0.0.0/0                         Up       0  Never
share       nemed~/acctShdw~back2             Up       0  Never

Counters last cleared: Never

prtlndA# ...
```

Clearing the Counters

All of the above reports show counters for state transitions, and two of them show additional heartbeat counters. You can clear these counters from priv-exec mode, using the clear counters redundancy command:

clear counters redundancy [heartbeat | transition | critical-services]

where **heartbeat | transition | critical-services** (optional) chooses only one type of counter to clear. By default, this command clears all counters. (Another option, clear counters redundancy network, was described earlier: *Clearing the Link-Transition Counters*, on page 3-55.)

heartbeat clears the heartbeat counters for the peer and the quorum disk.

transition clears the transition counters for both the peer and the quorum disk.

critical-services clears the counters in the critical-services report.

For example, this clears all heartbeat counters:

```
prtlndA# clear counters redundancy heartbeats
prtlndA# ...
```

This command clears all types of redundancy counters:

```
prtlndA# clear counters redundancy
prtlndA# ...
```

Showing the Ballots Cast for the Last Failover

Whenever a failover occurs, or when a failed switch comes back online, each member of the redundancy quorum casts a “ballot” to determine which peer should be senior. Every ballot has two components: the perceived senior switch and the perceived *epoch*. The epoch is an integer that increments every time a major redundancy-related event occurs, such as a failover. All members of the quorum — both peers and the quorum disk — have the same epoch number during normal operation; if one has a lower epoch number than the others, that member’s senior-switch vote is considered stale.

Use the show redundancy ballots command to see the most-recent ballots cast for seniority. Note that these are only ballots; the majority determines the senior switch:

```
show redundancy ballots
```

For example, this is a redundant pair where no failovers have occurred:

```
prtlndA# show redundancy ballots
          Ballot Cast
Node Switch/Quorum Disk   Senior Switch        Epoch
---- -------------------- -------------------- -----
*1   prtlndA              prtlndA              2
 2   prtlndB              prtlndA              2
 QD  192.168.74.83        -                    -
Last vote occurred at:  06:49:51 08/20/2010
prtlndA# ...
```

Showing the History

The redundancy processes record high-level log messages to indicate state changes during pair rendezvous and failovers. Use the show redundancy history command to see these log messages:

```
show redundancy history
```

For example:

```
prtlndA# show redundancy history

Date/Time(UTC) Recent History
--------------------------------------------------------------------------------
09-14 05:33:24 Quorum disk is online, system is ready for failover
09-14 05:33:19 Pair status changed from JoinWait to Formed
09-14 05:33:19 Peer switch 'prtlndB' is now online
09-14 05:33:07 Quorum disk 192.168.74.83:/exports/quorum-disk/portland1/ is online,
proceeding to verify rendezvous.
09-14 05:33:07 Quorum disk 192.168.74.83:/exports/quorum-disk/portland1 is now online.
09-14 05:33:01 Pair status changed from Rendezvous to JoinWait
09-14 05:33:01 Switch 'prtlndA' has a lower rendezvous IP address than 'prtlndB'.
[10.1.27.69, 10.1.23.11]
09-14 05:32:56 Cannot rendezvous because quorum disk
192.168.74.83:/exports/quorum-disk/portland1 is offline.
09-14 05:32:56 Pair status changed from Inactive to Rendezvous
09-14 04:21:55 Site quorum manager daemon started.

prtlndA# ...
```

Showing the Reboot History

To view the history of redundancy-related switch reboots, including reasons for the reboot, use the show redundancy reboot-history command:

```
show redundancy reboot-history
```

For example, this shows a system where users triggered two reboots and the redundancy software triggered two more:

```
prtlndA# show redundancy reboot-history

Version 5.01.000.11891 (Sep 10 2009 14:09:57) [jsmith]
Time of reboot: Mon Sep 14 00:10:07 2009
```

```
Message: user initiated reboot: admin at console

------------------

Version 5.00.005.11735 (Sep 10 2009 10:23:15) [admin12]
Time of reboot: Thu Sep 10 17:20:37 2009

Message: user initiated reboot: admin at 192.168.98.143

------------------

Version 2.03.000.09132 (Feb 12 2006 22:08:24) [jsmith]
Time of reboot: Wed Feb 22 11:54:26 2006
Message: REDUNDANCY requires this switch to reboot immediately.
Reason: Resource 'dme_svc-1' has failed/exited.

------------------

Version 2.03.000.09132 (Feb 12 2006 22:08:24) [jsmith]
Time of reboot: Mon Feb 20 17:08:25 2006
Message: Lost HA vlan connectivity and I am junior

prtlndA# ...
```

Showing All Redundancy Information

You can show all redundancy information with a single command:

`show redundancy all`

This shows all redundancy-related show commands concatenated together into one report, including show redundancy network (recall *Showing the Redundancy Network*, on page 3-55).

Recovering from Pairing Failures

The show commands may reveal that the enable failed and the pair did not form. In many cases, the show redundancy history command shows the configuration error and the recovery path is clear. Some more-abstract cases are documented in this section.

Clearing the Global Config from One Peer

The pair cannot form if namespaces and/or global servers are already defined on both peers. These are the most-important parts of the global-config, which is shared between redundant peers. This conflict appears in the show redundancy history output.

To correct this, choose one switch where the global config should be removed, then clear the global config from that switch. This removes all namespaces and global servers but preserves all subnets and interfaces.

Important

After it clears the global config, this command reboots the current switch.

After the switch reboots, the peers join automatically. They share the global config that was preserved.

From priv-exec mode, use the clear global-config command to remove all global-configuration parameters:

```
clear global-config
```

The CLI prompts for confirmation before clearing the config and rebooting. Enter **yes** to continue.

For example:

```
bstnA(cfg)# exit
bstnA# clear global-config
This removes all global configuration and reboots both chassis in the redundant pair in
order to ensure services are removed properly.

Are you sure? [yes/no] yes
bstnA#
Broadcast message (Fri Sep 24 13:53:15 2004):

The system is going down for reboot NOW!
...
```

Resolving a Private-Subnet Conflict

If the two chassis have matching private subnets, the pair cannot form. The show redundancy history command shows the following error in this case:

```
Cannot rendezvous with a switch whose private subnet matches mine. [a.b]
```

where *a* and *b* are the last two numbers in the matching subnet address.

To recover from this, use no enable to disable the redundancy configuration, then use ip private subnet reassign to choose a unique private subnet for one of the peers. The latter command reboots the chassis where you run it; recall *Changing the Private Subnet*, on page 3-12.

For example, the following command sequence notes the problem and then changes the private subnet on the local switch, "bstnB:"

```
bstnB(cfg)# show redundancy history

Date/Time(UTC) Recent History
--------------------------------------------------------------------------
08-05 15:26:55 Cannot rendezvous with a switch whose private subnet matches mine.
[150.192]
08-05 15:26:54 Pair status changed from Inactive to Rendezvous
08-05 14:26:13 Site quorum manager daemon started.


bstnB(cfg)# redundancy
bstnB(cfg-redundancy)# no enable
bstnB(cfg-redundancy)# exit
bstnB(cfg)# ip private subnet reassign

Reassign a new, unused, private subnet and reboot the chassis? [yes/no] yes
...
```

Then re-enable redundancy on both switches. See *Enabling Redundancy*, on page 7-20.

Promoting a Backup Switch to Active Status

A rare scenario causes a redundancy failure where the recovery mechanism is to promote the backup switch to active status. Consider a redundant pair where Peer A is active and Peer B is backup. Peer B fails, some time passes while Peer A runs alone, then Peer A fails unrecoverably. When Peer B recovers, it refuses to take an active role because it knows that it was down while Peer A was active: Peer A may therefore have changed its configuration or managed some transactions with back-end filers. If Peer B were to take control, all such configuration changes and transactions would be lost. In this scenario, Peer B waits indefinitely for Peer A and the redundant pair never comes back online.

You need to force Peer B to take an active role, thus abandoning any configuration changes or transactions that occurred on Peer A. The peer switch must be offline, and the current switch must have been up and waiting for its peer to come online for at least 5 minutes. Also, the quorum disk must be online and available.

From priv-exec mode, you can use the redundancy force-active command to force the current switch into the active role:

```
redundancy force-active
```

The CLI warns you about lost transactions and configuration that could result from this; enter **yes** to continue.

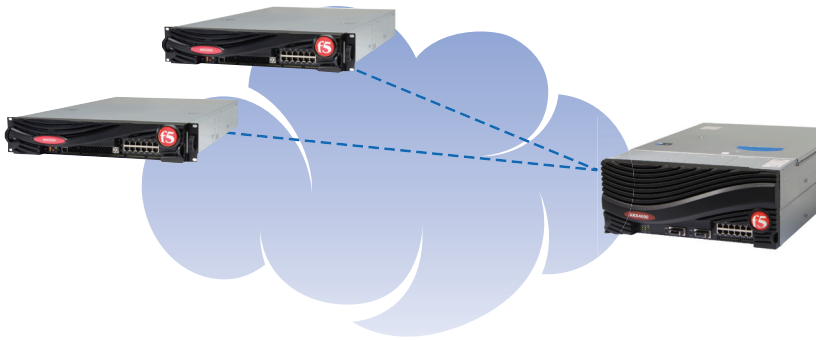
For example, the following command sequence forces “prtlndB” to take an active role in the redundant pair:

```
prtlndB(cfg-redundancy)# end
prtlndB# redundancy force-active
CAUTION: To avoid data corruption, the peer switch MUST BE OFFLINE and
remain offline while this command executes. Any global configuration
changes made on the Active peer switch while this switch was
unavailable will be lost. All namespaces will be re-imported
automatically to ensure consistency in the metadata. All in-flight
transactions not synchronized will be lost. NFS-based clients will
need to remount. Please contact Technical support for further advice.

Proceed? [yes/no] yes
prtlndB# ...
```

RON Tunnels to Redundant Pairs

RON tunnels do not fail over from one redundant peer to the other. If you create a RON tunnel from a remote switch to Peer A, you should create another tunnel from the same switch to Peer B. This ensures that the RON connection to the remote switch persists after a failover.



For example, the following command sequence creates two RON tunnels from an ARX-4000 to a redundant pair of ARX-2000s. The session starts with a tunnel to Peer A (named "prtlndA") and then configures the tunnel to Peer B ("prtlndB"):

Remote ARX-4000 -> Peer A

```
bstnA(cfg)# interface vlan 25
bstnA(cfg-if-vlan[25])# ip address 192.168.25.5 255.255.255.0
bstnA(cfg-if-vlan[25])# ron tunnel toPortland
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# peer address 192.168.74.66
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# no shutdown
bstnA(cfg-if-vlan-ron-tnl[25~toPortland])# exit
bstnA(cfg-if-vlan[25])# exit
bstnA(cfg)# ...
```

Peer A -> Remote ARX-4000

```
prtlndA(cfg)# interface vlan 74
prtlndA(cfg-if-vlan[74])# ip address 192.168.74.66 255.255.255.0
prtlndA(cfg-if-vlan[74])# ron tunnel toBoston
prtlndA(cfg-if-vlan-ron-tnl[74~toBoston])# peer address 192.168.25.5
prtlndA(cfg-if-vlan-ron-tnl[74~toBoston])# no shutdown
prtlndA(cfg-if-vlan-ron-tnl[74~toBoston])# exit
prtlndA(cfg-if-vlan[74])# exit
prtlndA(cfg)# ...
```

Remote ARX-4000 -> Peer B

```
bstnA(cfg)# interface vlan 25
; The IP address, 192.168.25.5, was configured above.
bstnA(cfg-if-vlan[25])# ron tunnel toPortland-B
bstnA(cfg-if-vlan-ron-tnl[25~toPortland-B])# peer address 192.168.74.96
bstnA(cfg-if-vlan-ron-tnl[25~toPortland-B])# no shutdown
bstnA(cfg-if-vlan-ron-tnl[25~toPortland-B])# exit
bstnA(cfg-if-vlan[25])# exit
bstnA(cfg)# ...
```

Peer B -> Remote ARX-4000

```
prtlndB(cfg)# interface vlan 74
prtlndB(cfg-vlan[74])# ip address 192.168.74.96 255.255.255.0
prtlndB(cfg-vlan[74])# ron tunnel toBoston
```

```
prtlndB(cfg-if-vlan-ron-tnl[74~toBoston])# peer address 192.168.25.5
prtlndB(cfg-if-vlan-ron-tnl[74~toBoston])# no shutdown
prtlndB(cfg-if-vlan-ron-tnl[74~toBoston])# exit
prtlndB(cfg-vlan[74])# exit
prtlndB(cfg)# ...
```

Failover Scenarios

The following situations result in an unplanned failover from the active peer to its redundant counterpart:

- Power failure - The ARX-2000 and ARX-4000 each have redundant power supplies, making this less likely.
- RAID failure - The ARX-2000 and ARX-4000 have redundant internal disks, making this less likely, too.
- Critical Service Loss - As explained above (recall *Showing the Critical Services*, on page 7-24), this only causes a failover if the standby peer has better access to its critical routes and shares.
- Unrecoverable hardware fault
- Unrecoverable software panic

NSM Redundancy

As mentioned previously, NSMs have an internal recovery mechanism to minimize the impact of core failures. This is independent of the ARX-to-ARX redundancy described earlier. This informational section describes the default configuration for all NSMs.

The NSM recovery feature is based on the redundancy between NSM processor cores within the same chassis, illustrated in this section. Most NSM processors contain two cores, where each core is assigned a *peer core* on another NSM processor. If one core fails, its peer takes the network traffic for both of them.

This section covers two related topics:

- *Redundancy Between NSM Cores*, which pertains to ARX-500, ARX-2000, and ARX-4000.
- *Redundancy Between NSM Daemons*, which pertains to ARX-1500, ARX-2500, and ARX-VE.

Redundancy Between NSM Cores

This section pertains to the following ARX models:

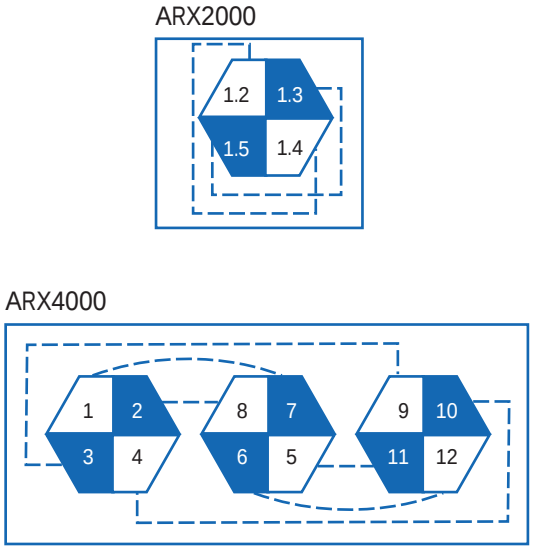
- ARX-500
- ARX-2000
- ARX-4000

Note

The ARX-500 contains a single core, and therefore does not support NSM core redundancy.

This figure shows the core-pairings for the ARX-2000 and the ARX-4000:

Figure 7.1 *Redundant NSM Core Pairings*



The physical NSM processors are represented by hexagons, each divided into two or more *sister cores* (1.2 and 1.3, 1.4 and 1.5, and so on). Each core is connected to a *peer core*, as shown by a dashed line; for example, 1.2 and 1.4 are peers in the ARX-2000, and 1 is peered with 7 in the ARX-4000. Peer cores are on separate physical processors whenever possible, to minimize the impact of a failed processor.

When a core fails, its physical processor reboots. All of the physical processor's cores then go into a "Standby" state (visible with the show processors command), ready to take traffic from their peer cores.

An ARX-2000 has a single physical processor, so that processor does not reboot unless both cores in a pair fail. For example, if core 1.3 fails, core 1.5 processes all of its traffic and no reboot occurs. If core 1.5 fails later, the physical processor reboots along with the rest of the chassis.

NSM Warm Restart

One aspect of NSM recovery is that once an NSM reanimates, not all cores within the same processor can be Up again without reloading; this is inherent to the design of the NSM's internal failover behavior. The system functions at a reduced level when this occurs. NSM warm restart functionality addresses this situation by restarting only the NSM core that failed. The other cores within the same processor remain unaffected. When restarted, the NSM comes back Up (not in Standby) and resumes its normal traffic load.

Whereas NSM recovery performs a cold reboot of all the processors in the NSM, including loading a new copy of the application image, NSM warm restart uses the in-memory copy of the image and restarts only the NSM core that crashed.

This feature is available on the ARX-2000 and ARX-4000. The ARX-500 contains a single core, and does not support this feature.

The warm restart mechanism is part of the NSM's exception processing. Exception types that are characterized as fatal (typically those caused by hardware) will not trigger warm restart. Non-fatal exception types (typically caused by software) generally will trigger warm restart, provided that nsm warm-restart is enabled in the ARX configuration and the number of restarts is less than a predetermined limit.

◆ Note

NSM warm restart functionality is disabled by default.

Execute the nsm warm restart CLI command in config mode to enable NSM warm restart. The no nsm warm restart CLI command disables the functionality.

You can check the state of NSM warm restart using the show nsm warm-restart CLI command.

The NSM restart history can be displayed using the show nsm warm-restart history CLI command. For example:

```
bstnA# show nsm warm-restart history
Proc           CPU          Restart            Date/Time(UTC)
                            Number
------------------------------------------------------------
1.2             A             1                06/22/2011 02:51:36 -0400
1.3             A             1                06/22/2011 02:52:00 -0400
1.4             B             0
1.5             B             0
Slot           CPU          Restart
                            Remaining
-----------------------------------
1               A             1
1               B             3
Restart Limit:      3
```

A processor can be specified explicitly as an argument to the command to display the warm restart history for that processor only; e.g., show nsm warm-restart history processor 1.2 .

Special Considerations For HA Configurations

NSM warm restart provides an additional level of high availability for the NSM. Peer NSMs replicate state between themselves constantly so that each NSM will be able to take over for the other, if necessary. When an NSM comes online, the state of both peers is synchronized.

When a restart occurs, the affected NSM is deprogrammed. When the NSM comes back up, it is it is programmed with its original configuration and possibly its peer's configuration if it had previously taken over for a failed peer.

Redundancy Between NSM Daemons

This section pertains to the following ARX models:

- ARX-1500
- ARX-2500
- ARX-VE

In these models, NSM cores are replaced by instances of a daemon named NSMD. The ARX-1500 runs two NSMDs, and the ARX-2500 runs four NSMDs. The ARX-VE runs one NSMD.

There are no local HA peers for these models; that is, there is no peering between daemons in the same unit.

One NSMD, the "master," listens on the VIP and load balances VIP connections. The other NSMDs are referred to as "workers." The client's IP address determines its target NSMD instance. The master NSMD sends file descriptors received on the VIP to the target worker NSMD via a UNIX domain socket.

If the NSMD process crashes:

1. It restarts automatically after the core dump.
2. While the NSMD restarts, other NSM processes will accept new client connection requests until the failed process restarts.
3. Once the failed NSM process has restarted, it is crucial that NFS clients connect to the same NSMD. During this startup process, the client requests for this NSMD process are queued to the master NSMD until the reanimation is complete. Once this process is started, all the connections from the master NSMD are handed off to the NSMD that served those clients originally.
4. If an NSM process fails too frequently, the chassis will reboot.

NSMD Reanimation

NSMD processes are started by the reanimator. When an NSMD process exits, the reanimator restarts it.

If an NSMD crashes, the clients served by that NSMD will be disconnected. As those clients reconnect, their new connections will be served by the replacement NSMD that was reanimated by the reanimator. The balance of client load across the available NSMDs is preserved by reanimation. There is no need to reboot an ARX-1500 or ARX-2500 after an NSMD crash to restore and redistribute client load among the NSMDs.

8

Configuring Management Access

Overview

You have strict control over management access to the ARX. You can customize the authentication service(s) for each management-access point in the system. A *management-access point* is a point of entry for an administrator:

- the serial CONSOLE port on the front panel,
- the Telnet server running on the out-of-band (MGMT) and in-band (VLAN) management interfaces,
- the SSH server running on the same interfaces,
- the HTTP/HTTPS server (for the GUI),
- the switch's SNMP agent,
- the HTTP-API/HTTPS-API server for the ARX API services.

For each of these access points, you can configure local authentication (through PAM), authentication through a remote RADIUS server, authentication through an Active Directory domain controller, through a combination of two or three authentication types, or you can deny access altogether.

By default, each management-access point is configured as follows:

- the CONSOLE port uses local authentication (as opposed to a remote RADIUS server or Active Directory domain controller),
- SSH and HTTPS also use local authentication and are enabled for all management interfaces,
- Telnet and HTTP are disabled,
- SNMP access is disabled, and
- HTTP-API and HTTPS-API are disabled.

Administrators can therefore log into the GUI over HTTPS (using any management interface), and they can log into the CLI over SSH (using any management interface or the CONSOLE). The less-secure HTTP, Telnet, and SNMP services are inaccessible by default. If these defaults are sufficient for your installation, you can skip ahead to *Tuning the SSH Service*, on page 8-28.

From cfg mode, use the management access command to change management access:

```
management access {console | telnet | ssh | http | http-api |
https | https-api | snmp | all}
```

where **{console | telnet | ssh | http | http-api | https | https-api | snmp | all}** is a required choice to select the management-access point. If you select **all**, you configure all management access points with the same settings.

This puts you into cfg-mgmt-access mode, where you permit or deny access to this management-access point. If you permit access, you also set the authentication service(s).

For example, the following command sequence begins configuring authentication for Telnet access to the management interfaces:

```
bstnA(cfg)# management access telnet
bstnA(cfg-mgmt-access[Telnet])# ...
```

Permitting Access

For Telnet, SSH, HTTP, HTTP-API, HTTPS, HTTPS-API, or SNMP management access, you can permit access through the out-of-band management interface, the in-band (VLAN) management interface, or both. This does not apply to console access, which is over a serial connection; skip to the next section if you are configuring management access for the console port only.

From cfg-mgmt-access mode, use the permit command to select the management interface(s) where management access is allowed:

permit {mgmt | vlan | all}

where

mgmt allows access through the out-of-band management interface (labeled MGMT on the front panel),

vlan allows access through any in-band (VLAN) management interface, and

all permits access through in-band or out-of-band interfaces.

For instructions on setting up out-of-band or in-band management, refer back to *Changing the Out-of-Band-Management Interface (optional)*, on page 4-18 or *Configuring an In-Band (VLAN) Management Interface*, on page 4-4.

For example, the following command sequence permits Telnet access through the out-of-band interface and all in-band interfaces:

```
bstnA(cfg)# management access telnet
bstnA(cfg-mgmt-access[Telnet])# permit all
bstnA(cfg-mgmt-access[Telnet])# ...
```

As another example, the following command sequence permits access to the SNMP agent through the out-of-band interface only:

```
bstnA(cfg)# management access snmp
bstnA(cfg-mgmt-access[SNMP])# permit mgmt
bstnA(cfg-mgmt-access[SNMP])# ...
```

Blocking Access

Use the no permit command to block management access:

no permit {mgmt | vlan | all}

where

mgmt disallows access through the out-of-band management interface (labeled MGMT on the front panel),

vlan prevents access via any in-band (VLAN) management interface, and

all prevents access through any management interface.

For example, the following command sequence stops SSH access through the out-of-band interface:

```
bstnA(cfg)# management access ssh
bstnA(cfg-mgmt-access[SSH])# no permit mgmt
bstnA(cfg-mgmt-access[SSH])# ...
```

Setting the Authentication Service

You can configure up to three authentication services, primary, secondary, and/or tertiary, for each management-access point. If the primary authentication service fails (for example, if a RADIUS server goes offline), the ARX falls back to the secondary service. Similarly, a failure of the secondary authentication service causes a fallback to the tertiary service. The tertiary authentication service is the last line of defense; if this fails too, access is denied.

Note

The SNMP agent does not support authentication services.

From cfg-mgmt-access mode, use the authentication command to set the primary, secondary, or tertiary authentication service:

```
authentication {primary | secondary | tertiary}
{active-directory | radius | local}
```

where

primary | secondary | tertiary sets the precedence of this authentication service, and

active-directory | radius | local is the authentication type:

- **active-directory** authenticates via the Active Directory domain controller on the network;
- **radius** authenticates via a RADIUS (Remote Authentication Dial-In User Service) server on the network, and
- **local** authenticates via services local to the ARX.

Important

For critical administrative user accounts, we recommend very strongly that you back up RADIUS and/or Active Directory authentication with local authentication to ensure that the management-access point does not become inaccessible. This is true particularly for the Console interface, which should always be available. If the RADIUS and/or Active Directory servers are offline and there is no local authentication to fall back upon, the management-access point is completely inaccessible.

When you include a user account in more than one authentication service, it is important to remember to keep the authentication credentials up to date in each service that is used. For example, if a password is changed in the primary authentication service but not in the secondary, it is possible for a user to log in with the old password. This is because the old password would

be accepted by the secondary authentication service even after being rejected by the primary. As a result of this, it is strongly recommended that you include each user account in one authentication service only, except for critical administrative users who should have local authentication also.

◆ Note

The ARX only considers a user's first 1000 Active Directory groups when using AD authentication. If the AD group needed for authentication is not within the first 1000 groups returned by the DC, the user will not be able to authenticate.

For example, the following command sequence configures RADIUS and local authentication for Telnet access:

```
bstnA(cfg)# management access telnet
bstnA(cfg-mgmt-access[Telnet])# authentication primary active-directory
bstnA(cfg-mgmt-access[Telnet])# authentication secondary radius
bstnA(cfg-mgmt-access[Telnet])# ...
```

Showing All Management Access

Use the show management access command to show all management-access settings for the ARX:

```
show management access
```

For example:

```
bstnA(cfg)# show management access


Service    Primary    Secondary    Tertiary    Allowed Interface
-------------------------------------------------------------
Console    Local      None         None        N/A
Telnet     AD         RADIUS       Local       VLAN/Management
SSH        AD         Local        None        VLAN/Management
HTTP       None       None         None        None
HTTPS      AD         Local        None        VLAN/Management
SNMP       N/A        N/A          N/A         VLAN/Management
HTTP-API   None       None         None        None
HTTPS-API  AD         Local        None        VLAN/Management

bstnA(cfg)# ...
```

This system supports both Telnet, SSH, and HTTPS access through the out-of-band (MGMT) interface as well as all in-band (VLAN) interfaces. Telnet uses Active Directory as its primary authentication service and RADIUS as its secondary; it falls back to local authentication if both of those services fail.

Removing an Authentication Service

From cfg-mgmt-access mode, use the no authentication command to remove the primary, secondary, or tertiary authentication service:

no authentication {primary | secondary | tertiary}

where **{primary | secondary | tertiary}** is a required choice; this is the authentication service to remove.

You can remove only the lowest-precedence service that is configured currently; that is, if a primary and secondary service are configured, you must remove the secondary service before you remove the primary one. If you remove all services, administrators are then denied access through the current management-access point.

For example, the following command sequence removes the secondary authentication service for SSH, then changes the primary service to "local" authentication:

```
bstnA(cfg)# show management access

Service    Primary    Secondary    Tertiary    Allowed Interface
-----------------------------------------------------------------
Console    Local      None         None        N/A
Telnet     AD         RADIUS       Local       VLAN/Management
SSH        AD         Local        None        VLAN/Management
HTTP       None       None         None        None
HTTPS      AD         Local        None        VLAN/Management
SNMP       N/A        N/A          N/A         VLAN/Management
HTTP-API   None       None         None        None
HTTPS-API  AD         Local        None        VLAN/Management

bstnA(cfg)# management access ssh
bstnA(cfg-mgmt-access[SSH])# no authentication secondary
bstnA(cfg-mgmt-access[SSH])# authentication primary local
bstnA(cfg-mgmt-access[SSH])# show management access

Service    Primary    Secondary    Tertiary    Allowed Interface
-----------------------------------------------------------------
Console    Local      None         None        N/A
Telnet     AD         RADIUS       Local       VLAN/Management
SSH        Local      None         None        VLAN/Management
HTTP       None       None         None        None
HTTPS      AD         Local        None        VLAN/Management
SNMP       N/A        N/A          N/A         VLAN/Management
HTTP-API   None       None         None        None
HTTPS-API  AD         Local        None        VLAN/Management

bstnA(cfg-mgmt-access[SSH])# ...
```

Configuring RADIUS

Remote Authentication Dial-In User Service (RADIUS) is an authentication service that uses an external server to perform client authentication. To use a RADIUS server, you must configure the server itself as well as the ARX. The RADIUS server must allow a connection from the ARX, and it must return a valid "group" attribute as part of a successful authentication.

The configuration examples in this section are specific to the FreeRadius implementation. Although the details of this configuration may differ from vendor to vendor, the general concepts apply to all RADIUS servers.

Allowing Client Access

The RADIUS server only accepts authentication requests from configured clients. Each proxy-IP address on the ARX is a potential client at the RADIUS server, so you must configure client access for all proxy IPs on the ARX. From the ARX CLI, use the show ip proxy-addresses command for a full list of proxy-IP addresses. See *Showing all Proxy IPs*, on page 4-8.

Setting the Secret Key

Each configured "client" on the radius server requires a *secret key*. This is a password to be entered at both the RADIUS server and the ARX (instructions for adding it to the ARX appear below). Each proxy IP requires its own client configuration; use the same secret key for all proxy IPs.

Sample Configuration: FreeRadius

At a FreeRadius server, you configure the clients in the following file:

/usr/local/etc/raddb/clients.conf

For every proxy IP address in the ARX, you would add one entry in the following format:

```
client proxy-ip-address {
secret = key
shortname = hostname
nastype = other
}
```

Where

proxy-ip-address is one proxy IP on the ARX.

key is an arbitrary shared secret that must be entered both here and at the ARX. (Instructions for entering this key at the ARX appear below.)

hostname can be found with the CLI show hostname command at the ARX.

For example, the following configuration file configures six proxy IPs with the same shared secret and hostname:

```
...
client 192.168.25.31 {
        secret = $3cretPa$$w0rd
        shortname = bstnA
        nastype = other
}
client 192.168.25.32 {
        secret = $3cretPa$$w0rd
        shortname = bstnA
        nastype = other
```

```
}
client 192.168.25.33 {
        secret = $3cretPa$$w0rd
        shortname = bstnA
        nastype = other
}
client 192.168.25.34 {
        secret = $3cretPa$$w0rd
        shortname = bstnA
        nastype = other
}
client 192.168.25.141 {
        secret = $3cretPa$$w0rd
        shortname = bstnA
        nastype = other
}
client 192.168.25.142 {
        secret = $3cretPa$$w0rd
        shortname = bstnA
        nastype = other
}
...
```

Mapping Users to Groups

The next step in configuring a RADIUS server is mapping each administrative *user* to a valid *group*. A user's group determines what actions that user is allowed perform on the ARX. (Refer back to *Adding the User to a Group*, on page 2-6.) When the ARX is configured for RADIUS, it depends on the RADIUS server to associate administrative users with their groups.

For each administrative user configured on the RADIUS server, use the 'Cisco-Account-Info' attribute to define the user's group:

`Cisco-Account-Info = "G`*`group-name`*`"`

where *group-name* is the name of a group configured on the ARX. From the ARX CLI, use show group all for a list of valid groups; see *Listing All Groups*, on page 2-5.

Note the "G" that appears before the group name. This is required to establish the parameter as a Group name.

Different RADIUS vendors handle user and attribute management differently, but the fundamental requirement is that the RADIUS server returns the 'Cisco-Account-Info' attribute as part of the authentication response.

Sample Configuration: FreeRadius

On a FreeRadius server, you can configure administrative users by editing the following configuration file:

/usr/local/etc/raddb/users

For example, the following entry defines the “crypto-officer” group for the “admin” user:

```
admin
    Cisco-Account-Info = "Gcrypto-officer"
```

Using an Existing User Configuration

Some sites may have existing RADIUS configurations that use the ‘Cisco-Account-Info’ attribute to map a group to the “admin” user. In these cases, it is unlikely the group name will be a valid F5 group. To allow the ARX to use the existing group name, use the group command to configure that group name on the ARX. See *Adding a Group (optional)*, on page 2-12 for full configuration instructions.

For example, suppose a RADIUS server defines an administrative group called “tier3” which is supposed to have the highest access privileges possible:

```
admin
    Cisco-Account-Info = "Gtier3",
...
```

At the ARX, the following command sequence creates a new group named “tier3” and provides the group with the crypto-officer role. The crypto-officer role has the highest possible privileges for accessing the CLI.

```
bstnA(gbl)# group tier3
bstnA(gbl-group[tier3])# role crypto-officer
bstnA(gbl-group[tier3])# exit
bstnA(gbl)# ...
```

Configuring a RADIUS Server at the ARX

You perform the remaining steps at the ARX’s CLI. From gbl mode, use the radius-server command to identify a RADIUS server:

radius-server *hostname-or-ip-address*

where *hostname-or-ip-address* (1-128 characters, or an IP address) identifies the RADIUS server. You must configure DNS lookups before you use a hostname; see *Configuring DNS Lookups*, on page 4-31.

This places you into gbl-radius mode, where you must set the shared-secret that was configured at the RADIUS server. You can also edit the port used for RADIUS communication, the number of milliseconds for a timeout, and the number of connect-failure retries.

For example, the following command set starts configuring a RADIUS server at 192.168.25.201:

```
bstnA(gbl)# radius-server 192.168.25.201
bstnA(gbl-radius[192.168.25.201])# ...
```

Configuring Multiple RADIUS Servers

You can configure multiple RADIUS servers to be used by the ARX. If the first server fails, the ARX tries the next one, and so on. The switch follows the order in which you entered the servers. If all servers fail, the RADIUS service as a whole fails.

Setting the Shared-Secret Key

The only required step in configuring a RADIUS server is setting its key. The ARX and the RADIUS server must have identical keys (shared-secret strings) in order to perform authentication. From gbl-radius mode, use the key command to set the key:

key

The CLI then challenges you for the key, then you must validate the key. For example, the following commands use the "$3cretPa$$w0rd" key:

```
bstnA(gbl)# radius-server 192.168.25.201
bstnA(gbl-radius[192.168.25.201])# key
Key: $3cretPa$$w0rd
Validate Key: $3cretPa$$w0rd
bstnA(gbl-radius[192.168.25.201])# ...
```

Once the key is set and you exit gbl-radius mode, management services can authenticate administrators using the RADIUS server. You have the option to reset other variables, described below.

Erasing the Key

By erasing the key, you disable authentication at the current RADIUS server. From gbl-radius mode, use the no key command to erase the shared-secret key:

no key

For example:

```
bstnA(gbl)# radius-server 192.168.25.207
bstnA(gbl-radius[192.168.25.207])# no key
bstnA(gbl-radius[192.168.25.207])# ...
```

Changing the Authentication Port (optional)

By default, the ARX sends its RADIUS packets to port 1812, the port number officially assigned for RADIUS (see RFC 2138). If your RADIUS server listens at a different port, you can change it. From gbl-radius mode, use the auth-port command to reset the port number:

auth-port *port-number*

where *port-number* (1024-65535) is the new port number to use for communication with the RADIUS server.

For example, the following commands change the port number to 5555 for the RADIUS server at 192.168.25.207:

```
bstnA(gbl)# radius-server 192.168.25.207
bstnA(gbl-radius[192.168.25.207])# auth-port 5555
bstnA(gbl-radius[192.168.25.207])# ...
```

Reverting to the Default Port

Use the no auth-port command to reset the port to the default, 1812:

```
no auth-port
```

For example:

```
bstnA(gbl)# radius-server 192.168.25.201
bstnA(gbl-radius[192.168.25.201])# no auth-port
bstnA(gbl-radius[192.168.25.201])# ...
```

Changing the Timeout (optional)

By default, the ARX times out after trying to connect for 3 milliseconds. After the timeout expires, the switch retries its request as many times as specified with the retries command (described below). If all retries fail, the request fails. A failed request causes a fallback to the next level of authentication service (secondary or tertiary), if one is configured.

From gbl-radius mode, use the timeout command to change the timeout:

```
timeout milliseconds
```

where *milliseconds* (3-65535) is the number of milliseconds before timing out.

For example, the following commands change the time-out to 10 milliseconds for the RADIUS server at 192.168.25.207:

```
bstnA(gbl)# radius-server 192.168.25.207
bstnA(gbl-radius[192.168.25.207])# timeout 10
bstnA(gbl-radius[192.168.25.207])# ...
```

Reverting to the Default Timeout

Use the no timeout command to revert to the default, 3 (milliseconds):

```
no timeout
```

For example:

```
bstnA(gbl)# radius-server 192.168.25.201
bstnA(gbl-radius[192.168.25.201])# no timeout
bstnA(gbl-radius[192.168.25.201])# ...
```

Changing the Number of Retries (optional)

By default, the ARX retries a failed connect 3 times before giving up. If all retries fail, the request fails. A failed request causes a fallback to the next level of authentication service (secondary or tertiary), if one is configured.

From gbl-radius mode, use the retries command to change the number of retries:

```
retries number
```

where *number* (3-65535) is the number of retries.

For example, the following commands reduce the number of retries for the RADIUS server at 192.168.25.207:

```
bstnA(gbl)# radius-server 192.168.25.207
bstnA(gbl-radius[192.168.25.207])# retries 4
bstnA(gbl-radius[192.168.25.207])# ...
```

Reverting to the Default Retries

Use the no retries command to revert to the default, 3:

```
no retries
```

For example:

```
bstnA(gbl)# radius-server 192.168.25.201
bstnA(gbl-radius[192.168.25.201])# no retries
bstnA(gbl-radius[192.168.25.201])# ...
```

Showing the RADIUS Configuration

Use the show radius-server command to view the current configuration for RADIUS:

```
show radius-server
```

If you have configured multiple RADIUS servers, they appear in the order they were configured. This is the same order in which the switch uses them; if the first server fails, the switch falls back to the second, and so on.

For example:

```
bstnA(gbl)# show radius-server

Hostname                        Authport  Acctport  Timeout  Retries
-------------------------------------------------------------------
192.168.25.201                  1812      1813      3        3
192.168.25.207                  5555      1813      10       4

bstnA(gbl)# ...
```

Removing the RADIUS Configuration

From gbl mode, use the no radius-server command to remove the configuration for a RADIUS server:

```
no radius-server hostname-or-ip-address
```

where *hostname-or-ip-address* (1-128 characters, or an IP address) identifies the RADIUS server to remove.

WARNING

If this RADIUS server provides the only authentication for a management-access point, this command disables that management access. For example, if RADIUS is the only authentication service for SSH management, no one can use SSH to access the CLI.

Use the show management access *command to see the mapping of management services (such as SSH and Telnet) to authentication services; see Showing All Management Access, on page 8-7.*

For example, the following command removes the configuration for the RADIUS server at 192.168.25.207:

```
bstnA(gbl)# no radius-server 192.168.25.207
bstnA(gbl)# ...
```

Configuring Active Directory Authentication

In order to use Active Directory authentication on the ARX, you need to configure two aspects of the global configuration. These tasks are similar to configuration procedures that must be executed when configuring CIFS support; however, CIFS support involves a number of additional considerations related to filers that are not described here.

These tasks are:

- *Configuring a Proxy User*
- *Configuring a Seed Domain*

Configuring a Proxy User

In order to access an Active Directory domain controller, the ARX must be configured with a proxy user with Windows credentials. A proxy user is a single username/password combination in a particular Windows domain that the ARX uses to access the Active Directory domain controller. The proxy user enables the ARX to query the Windows Active Directory for its Windows Domains and domain controller (DC) addresses.

If you later configure a namespace in the same Windows domain, you can apply this proxy user configuration to that namespace. You can apply the same proxy user to multiple namespaces in the same domain.

The global configuration mode command proxy-user creates the proxy user and puts the CLI into gbl-proxy-user mode, from which the proxy user is defined.

In gbl mode, create a proxy user:

proxy-user ***name***

where ***name*** (1-32 characters) is a name you specify.

This enters gbl-proxy-user mode, where you specify the Windows domain, username, and password for the proxy user.

For example, the following command sequence creates a proxy user named "acoProxy2:"

```
bstnA(gbl)# proxy-user acoProxy2
bstnA(gbl-proxy-user[acoProxy2])# ...
```

Continue in gbl-proxy-user mode to define the proxy user's characteristics:

- Window domain
- username/password combination
- description (optional)

Specifying the Windows Domain

The first step in configuring a proxy user is to specify its Windows domain. From gbl-proxy-user mode, use the windows-domain command to specify the domain:

windows-domain *name*

where ***name*** is 1-64 characters. Enter a full FQDN (such as "myco.com") for the ARX to use Kerberos for its proxy-user authentications. The ARX uses the first part of the FQDN (such as "myco", above) if it requires a NetBIOS-style name.

For example, the following command sequence specifies the "MEDARCH.ORG" domain for the "acoProxy2" proxy user and another domain for the "acoProxy3" proxy user:

```
bstnA(gbl)# proxy-user acoProxy2
bstnA(gbl-proxy-user[acoProxy2])# windows-domain MEDARCH.ORG
bstnA(gbl-proxy-user[acoProxy2])# exit
bstnA(gbl)# proxy-user acoProxy3
bstnA(gbl-proxy-user[acoProxy3])# windows-domain FDTESTNET.COM
bstnA(gbl-proxy-user[acoProxy3])# ...
```

Using a Pre-Windows-2000 Domain Name

Some back-end filers cannot accept an FQDN (such as "FDTESTNET.COM" in the example above) for the ARX's NTLM authentications. These filers accept only a domain name format used prior to the release of Windows 2000: 1-15 bytes and no periods (“.”). All filers accept this old-style domain format, known to the ARX as "pre-win2k," so the ARX performs its NTLM authentications with the first part of the FQDN you typed above. This is the name before the first period (“.”) in the FQDN, up to 15 characters. In the example above, this would be "FDTESTNET."

The pre-win2k name is used when the environment does not use Active Directory, or does not support Kerberos authentication. For example, the client may be a computer using a pre-win2k version of Windows, or some clients that are joined to the domain may not have a forest trust with the same forest to which the ARX CIFS service and filers are joined.

If the pre-win2k domain name is not configured explicitly, the pre-win2k domain name is discovered automatically during AD configuration and/or AD discovery. In the unlikely event that a pre-win2k domain name is not identified at that time, the ARX will derive a pre-win2k domain name as a last resort. This is done by truncating the FQDN to the first 15 characters before the first period.

For a domain that uses a different pre-Windows-2000 name, you can use the optional pre-win2k-name argument to configure a pre-win2k domain name explicitly, but this should not be necessary if Active Directory is in use:

windows-domain *name* pre-win2k-name *old-style-name*

where ***old-style-name*** is 1-15 characters and does not contain any periods.

For example, the following command sequence enters "BOSTONCIFS," a pre-Windows-2000 domain name, for the "acoProxy3" proxy user:

```
bstnA(gbl)# proxy-user acoProxy3
bstnA(gbl-proxy-user[acoProxy3])# windows-domain FDTESTNET.COM pre-win2k-name BOSTONCIFS
bstnA(gbl-proxy-user[acoProxy3])# ...
```

The ARX will use available pre-win2k domain names in the following precedence order:

1. A pre-win2k domain name configured explicitly using the pre-win2k-name option. This should not be necessary.
2. A pre-win2k domain name discovered automatically during AD configuration and/or AD discovery. This is the default behavior, and should work reliably for most implementations.
3. A pre-win2k domain name derived by truncating the FQDN to the first 15 characters before the first period. The ARX does this as a last resort.

Specifying the Username and Password

The final required step in configuring a proxy user is to specify a username and password. The username must be for a user in the corresponding Windows domain. If you use *local* groups in a Windows cluster, add the user to the local group at every node in the cluster.

Use a domain-level user account (not a local one) if any of your Windows filers support User Account Control (UAC). UAC is a security feature introduced with Windows Server 2008.

From gbl-proxy-user mode, use the user command to specify the username:

user *username*

where ***username*** (1-64 characters) is a valid username in the proxy-user's domain.

The CLI prompts you for the user's password, then prompts to validate the password.

For example, the following command sequence specifies the username, "jqpublic:"

```
bstnA(gbl-proxy-user[acoProxy2])# user jqpublic
Password: jqpasswd
Validate Password: jqpasswd
bstnA(gbl-proxy-user[acoProxy2])# ...
```

Adding a Description

You can add a description to the proxy-user configuration, for use in the show command below. The description can differentiate the proxy user from others. From gbl-proxy-user mode, use the description command to describe the proxy user:

description *text*

where ***text*** is 1-255 characters. Quote the text if it contains any spaces.

For example:

```
bstnA(gbl)# proxy-user acoProxy2
bstnA(gbl-proxy-user[acoProxy2])# description "user with backup and admin creds on our servers"
bstnA(gbl-proxy-user[acoProxy2])# ...
```

Removing the Description

From gbl-proxy-user mode, use no description to remove the description string:

```
no description
```

For example:

```
bstnA(gbl)# proxy-user acoProxy3
bstnA(gbl-proxy-user[acoProxy3])# no description
bstnA(gbl-proxy-user[acoProxy3])# ...
```

Listing All Proxy Users

You can use the show proxy-user command to get a list of all proxy users on the ARX:

```
show proxy-user
```

For example:

```
bstnA(gbl)# show proxy-user

Name             Windows Domain                Pre-Win2k        User Name
                 Description
-------------------------------------------------------------------------------
acoProxy1        WWMEDNET.COM                  WWMEDNET         jqprivate
                 jq's admin account

acoProxy3        FDTESTNET.COM                 BOSTONCIFS       jqtester

cifs_admin       MEDARCH.ORG                   MEDARCH          Administrator

nas_admin                                                       root

ny_admin         NY.COM                        NY               jqpublic

acoProxy2        MEDARCH.ORG                   MEDARCH          jqpublic
                 user with backup and admin creds on our servers

bstnA(gbl)# ...
```

Showing One Proxy User

To focus on one proxy user, you can specify a name in the show proxy-user command:

```
show proxy-user name
```

where ***name*** (1-32 characters) identifies the proxy user to show.

For example:

```
bstnA(gbl)# show proxy-user acoProxy2

Name             Windows Domain                Pre-Win2k        User Name
                 Description
-------------------------------------------------------------------------------
acoProxy2        MEDARCH.ORG                   MEDARCH          jqpublic
                 user with backup and admin creds on our servers

bstnA(gbl)# ...
```

Removing a Proxy User

From gbl mode, use no proxy-user to remove a proxy-user configuration:

```
no proxy-user name
```

where ***name*** (1-32 characters) identifies the proxy user to remove.

For example, the following command sequence removes a proxy user called proxyNYC:

```
bstnA(gbl)# no proxy-user proxyNYC
```

Configuring a Seed Domain

In order to be able to use Active Directory for ARX authentication, you must first identify the *Active Directory (AD) forest* in your Windows network. Given one of the domains in an Active Directory forest, called the *seed domain*, the ARX can automatically discover all of the domains and DCs in the forest. The discovery operation performs a DNS lookup to find the seed domain's DC, then queries that DC for the names of other DCs in the same forest. The DC uses FQDNs instead of IP addresses; the ARX must perform more DNS lookups to translate those names into IP addresses.

◆ Note

Before you start discovering AD forests, configure the switch to perform DNS lookups at a properly-configured DNS server; refer to *Configuring DNS Lookups, on page 4-31.* *Choose a DNS server that can perform lookups for all of the domains in the desired forest.*

Once you have DNS configured, you can use the active-directory update command to automatically discover the AD forest. This command is in priv-exec mode:

```
active-directory update seed-domain domain proxy-user proxy
[domain-controllers max-dcs] [site-name site] [verbose] [tentative]
```

where

domain (1-255 characters) is one domain in the forest. The ARX uses this domain name as the name of the AD-forest object.

proxy (1-32 characters) is the name of a proxy-user configuration (recall *Configuring a Proxy User*, on page 8-15). The CLI uses the proxy-user credentials to query the ***domain***'s DC for other DCs in the AD forest. The credentials should belong to a domain in the same AD forest or a trusted forest. Use show proxy-user for a full list of all proxy users and their domains.

max-dcs (optional, 1-100) is the maximum number of DCs per domain. If the DNS lookup for a domain yields a larger number of DCs, the ARX uses the priority set by the DNS server. For example, if the ***max-dcs*** setting is 3 and the DNS server returns 15 DCs, the ARX selects the first 3 DCs from the list. If you omit this, the ARX uses all DCs provided by the DNS server.

site (optional, 1-64 characters) Use this option if the AD's site configuration does not include the proxy-IP subnet. This identifies the AD site for the ARX. If the ARX knows of multiple DCs that can answer the same query, it prefers DCs in its own site (if there are any) over DCs in any other site. The site name is defined on a DC with the Active Directory Sites and Services plugin. The site name is case insensitive, so "boston" and "BOSTON" are equivalent. If you omit this option, the ARX software uses the AD site configured for the *ip proxy-address* subnet.

verbose (optional) causes the CLI to display the results of the AD-forest discovery as the operation progresses, followed by a summary of the AD-forest changes in the ARX configuration.

tentative (optional) causes the ARX to discover the AD forests domains and DCs, but prevents it from adding the AD forest to its configuration.

This command generates a report with details about the discovery process. This includes all of the information that you see when you use the **verbose** flag. The CLI displays the name of the report after you issue the command. Use show reports *report-name* to read the report.

For example, the following command automatically discovers the "MEDARCH.ORG" forest:

```
bstnA# active-directory update seed-domain medarch.org proxy-user acoProxy2

Report File : active-directory-MEDARCH.ORG.rpt
bstnA# . . .
```

Configuring API Access

Activate the ARX API using the management access command followed by the permit command. This opens a port where a SOAP client can send queries to this volume. The HTTP-API port is 83, and the HTTPS-API port is 843.

Use the following URL syntax to access the API documentation via HTTP:

http://*arx-management-ip*:83/arx-api/

where *arx-management-ip* is either the out-of-band management IP address or an in-band (VLAN) management IP address. The permit command determines which of these address types are available for access. Use show interface mgmt and/or show interface vlan to find the IP addresses for each interface.

This URL accesses the same documentation through HTTPS:

https://*arx-management-ip*:843/arx-api/

where *arx-management-ip* is a valid management IP address.

Use the show statistics api command to find usage statistics for the API.

Showing All Management Sessions

Use the show sessions command to view all active-management sessions on the ARX:

show sessions

The report shows one line for each session. An asterisk (*) at the beginning of the line indicates the current management session. The ID is a handle for identifying the session; you can use it in the clear session command (below) to forcibly log the administrator off. The Username identifies the administrative user. The Line Type is equivalent to the management-access point (ssh, telnet, console, or unknown). The Connect Time shows, in minutes and hours, how long the session has been running (if less than a minute, no data), and the Source IP is the IP address of the remote management station.

For example, the following switch has one active login session:

```
bstnA# show sessions

Connected Sessions
------------------

Session ID:                  30982 (local session)
Username:                    admin
Access:                      ssh
Connect Time:                0 days, 00:01:02
Source IP:                   172.16.100.183

bstnA# ...
```

Clearing a Management Session

As mentioned above, you can use the clear session command to stop a management session. Use this command from priv-exec mode:

clear session *session-id*

where ***session-id*** (1-128 characters) identifies the management session to disconnect. You can get this ID from the show sessions command (above).

This logs off the session. If a user is logged in when you do this, they must re-enter their username and password to re-connect.

For example, the following command sequence exits to priv-exec mode, shows all connected sessions, clears the telnet session (67106), then shows that it cleared:

```
bstnA(cfg)# end
bstnA# show sessions

Connected Sessions
------------------

Session ID:                  30982 (local session)
Username:                    admin
Access:                      ssh
```

```
Connect Time:                 0 days, 00:01:02
Source IP:                    172.16.100.183

Session ID:                   67106 (local session)
Username:                     admin
Access:                       telnet
Connect Time:                 1 days, 00:08:33
Source IP:                    172.16.152.19

Session ID:                   73441 (local session)
Username:                     admin
Access:                       ssh
Connect Time:                 0 days, 00:01:37
Source IP:                    10.1.1.56


bstnA# clear session 67106
bstnA# show sessions

Session ID:                   30982 (local session)
Username:                     admin
Access:                       ssh
Connect Time:                 0 days, 00:01:02
Source IP:                    172.16.100.183

Session ID:                   73441 (local session)
Username:                     admin
Access:                       ssh
Connect Time:                 0 days, 00:01:37
Source IP:                    10.1.1.56

bstnA# ...
```

Showing Authentication Statistics

The ARX has counters for all authentication access. From any mode, use the show statistics authentication command to see these counters:

```
show statistics authentication
```

The report shows the number of successful accesses and the number of failures. A failure is a rejected username and password.

All counters are reset on reboot, or for a CLI command (below).

For example, the following switch has had 8 successful logins through SSH and one failed login through the CONSOLE port:

```
bstnA(cfg)# show statistics authentication


                           Success   Failure
                           -------   -------
Telnet
  Local database                 0         0
  Active Directory               0         0
  RADIUS                         0         0

SSH
  Local database                29         0
  Active Directory               0         0
```

```
  RADIUS                          0         0

Console
  Local database                  0         0
  Active Directory                0         0
  RADIUS                          0         0

API
  Local database                  0         0
  Active Directory                0         0
  RADIUS                          0         0

HTTP/HTTPS
  Local database                  0         0
  Active Directory                0         0
  RADIUS                          0         0

Totals                            29        0

bstnA(cfg)# ...
```

Clearing Authentication Counters

From priv-exec mode, use the clear statistics authentication command to clear all authentication-access counters:

```
clear statistics authentication
```

This resets all of the counters in show statistics authentication to zero.

For example, the following command sequence exits to priv-exec mode, clears all authentication counters, then shows the cleared statistics:

```
bstnA(cfg)# end
bstnA# clear statistics authentication
bstnA# show statistics authentication

                          Success   Failure
                          -------   -------
Telnet
  Local database                0         0
  Active Directory              0         0
  RADIUS                        0         0

SSH
  Local database                0        0
  Active Directory              0         0
  RADIUS                        0         0

Console
  Local database                0         0
  Active Directory              0         0
  RADIUS                        0         0

API
  Local database                0         0
  Active Directory              0         0
  RADIUS                        0         0

HTTP/HTTPS
  Local database                0         0
  Active Directory              0         0
```

```
  RADIUS                          0          0
Totals                            0          0

bstnA# ...
```

Sample - Configuring Authentication

The following sample script configures authentication for CLI users. The primary authentication for Telnet is through RADIUS, where the RADIUS server is at 192.168.25.201. The secondary authentication is done through local authentication services. SNMP is configured to be accessible only through the out-of-band (MGMT) management interface. All other management access is left in its default configuration.

```
config

management access telnet
permit all
authentication primary radius
authentication secondary local
exit

management access snmp
permit mgmt
exit
exit

; Set up the RADIUS server. Presume the server itself is
; properly configured.
global

radius-server 192.168.25.201
key
Key: $3cretPa$$w0rd
Validate Key: $3cretPa$$w0rd
exit
exit
```

Removing Management Access

You can remove a management-access configuration, disabling that form of management access. From cfg mode, use the no management access command:

```
no management access {console | telnet | ssh | http | http-api |
https | https-api | snmp}
```

where

console disables CLI access at the CONSOLE port,

telnet disables Telnet access to the management interfaces,

ssh disables SSH (Secure SHell) access to management interfaces,

http disables HTTP (non-secure) access to management interfaces,

http-api disables HTTP (non-secure) access to API services,

https disables HTTPS (secure) access to management interfaces,

https-api disables HTTPS (secure) access to API servicess,

snmp stops access to the SNMP agent.

Important

If you disable all management access, no one can log into the switch. This requires support from F5's Customer-Service personnel to regain entry to the box.

For example, the following command sequence shuts down all SNMP access:

```
bstnA(cfg)# no management access snmp
bstnA(cfg)# ...
```

Tuning the SSH Service

SSH authenticates administrators who access the CLI. This is one of the authentication services you may have enabled or disabled above; by default, SSH is enabled for all management interfaces on the switch. If you have disabled SSH for all management-access points, you can skip this section.

Enabling SSHv1

By default, the SSH service supports SSHv2 only. SSHv1 has well-known security holes and is not recommended for management access. However, some management stations do not support SSHv2. To accommodate these management stations, use the ssh-v1 enable command from cfg mode:

```
ssh-v1 enable
```

For example:

```
bstnA(cfg)# ssh-v1 enable
bstnA(cfg)# ...
```

Disabling SSHv1

We recommend disabling SSHv1 if possible. Use no ssh-v1 enable to drop SSHv1 support and rely only on the more-secure SSHv2:

```
no ssh-v1 enable
```

For example:

```
bstnA(cfg)# no ssh-v1 enable
bstnA(cfg)# ...
```

Regenerating the Switch's SSH Keys

For added security, you can occasionally regenerate the switch's public/private key pairs. SSH clients use a public key as the identity of the ARX; by changing the switch's keys periodically, you make it more difficult for an attacker to pose as the switch. After generating the host-key pairs, you must redistribute the public keys to all of the switch's SSH-client machines (as described below).

From priv-exec mode, use the ssh-host-key generate command to regenerate the host keys at the ARX.

```
ssh-host-key generate
```

The CLI prompts you that remote SSH clients must all reinstall their keys for the ARX; enter **yes** to proceed.

For example:

```
bstnA(cfg)# end
bstnA# ssh-host-key generate
Warning: this command replaces the current SSH host keys with new key material. Host
public keys must be redistributed to SSH clients following this action. Proceed? [yes/no]
yes
```

```
bstnA# ...
```

Important

Do not exit the CLI until you read the next section. The new host key must be installed on at least one SSH client to log back in through SSH. The section below describes how to display the key before exiting the CLI.

Showing and Distributing the New Keys

After regenerating the switch's public/private key pairs, clients cannot easily access the switch through SSH until one of the new public keys is installed at each client machine. Before you exit the CLI, you output the public host keys. Then you copy and paste the right key into an SSH client's configuration.

To show the public host keys on the switch, use the show ssh-host-key command from any mode:

show ssh-host-key [dsa | rsa | rsa1]

where

dsa shows only the DSA (over SSHv2) key,

rsa focuses in the RSA (over SSHv2) key, and

rsa1 shows only the RSA over SSHv1 key.

By default, this command shows all three host keys.

You can copy the appropriate host key and paste it into an SSH configuration file at a client machine. (Consult your SSH-client documentation for guidance on the correct key type and configuration file.) Then SSH users can reconnect from the client machine to the ARX.

This command sequence continues the previous example, where all host keys were regenerated. This sequence shows the RSA key at the ARX, logs off, tries and fails to re-establish an SSH connection, then installs the key on the Linux client. Once the key is installed, the SSH connection succeeds. The host key is associated with the in-band (VLAN) management address for the switch, 192.168.25.5:

```
bstnA# show ssh-host-key rsa

ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAtB123IvN5D+nLB2eBH1XpC+OC+zvZpO/4v4nVwgX1MaJoadWVIGSGp2rfO+rnE
uh3UBNWP29uX1TWUqpyVqKYQLp/XHR7T6GYNnE3B4ACK58duLSZ0c6kVIutsGWezpduvCjDhLCcm47V1506yhSSLAR
Dvu4fPoG6r7zP1+TPK0=

bstnA# exit
Connection to 192.168.25.5 closed.
juser@clientLinux:/$
```

This closes the connection to the ARX. An attempt to reconnect fails:

```
juser@clientLinux:/$ ssh admin@192.168.25.5
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
```

```
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
1f:d6:cf:43:d6:33:18:13:44:6d:8a:e9:b5:9f:81:cf.
Please contact your system administrator.
Add correct host key in /home/juser/.ssh/known_hosts to get rid of this message.
Offending key in /home/juser/.ssh/known_hosts:60
RSA host key for 192.168.25.5 has changed and you have requested strict checking.
Host key verification failed.
juser@clientLinux:/$ cd ~/.ssh
juser@clientLinux:~/.ssh$
```

As instructed by the prompts, this next command pastes the RSA host key (from the show command, above) into the user's .ssh/known_hosts file. If there is an old RSA-ky entry for the host, replace it. For example:

```
juser@clientLinux:~/.ssh$ vi known_hosts
172.16.25.16 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA0opdS17hhVpgGGFOtsVjGnRlD0ntTra/A6IgJcSbA2J2xeXXknLwDNZSWtQsVo
ZGFw73VSf2sfhVL6W2Y4MeRlgwZfTmK7dY2ZAGtDrF1uZQ1jALKdSzp34pPy+it9pdaUQzOdKzz946KDd6LuZj2O2E
KY+gKmo/+4KxYQ08iqs=
...
192.168.25.5 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAtB123IvN5D+nLB2eBH1XpC+OC+zvZpO/4v4nVwgX1MaJoadWVIGSGp2rfO+rnE
uh3UBNWP29uX1TWUqpyVqKYQLp/XHR7T6GYNnE3B4ACK58duLSZ0c6kVIutsGWezpduvCjDhLCcm47V1506yhSSLAR
Dvu4fPoG6r7zP1+TPK0=
```

The next SSH connection succeeds:

```
juser@clientLinux:~/.ssh$ ssh admin@192.168.25.5
Password: myPa$$wd
bstnA>
```

Manually Setting the Host Key

Some sites actively manage their host-key pairs with an external SSH application, such as PuTTYgen (on Windows) or ssh-keygen (on Unix). You can retrieve a private host key from such an application and apply it as a private host key for the ARX. From cfg mode, use the ssh-host-key command to install an existing SSH-host key onto the ARX:

```
ssh-host-key {rsa | dsa}
```

where **rsa | dsa** chooses the SSHv2 key type.

The CLI prompts you for the SSH host key. Paste the private key at the prompt. The private key must start with "-----BEGIN RSA PRIVATE KEY-----" and end with a similar line, and must have a <Return> character at the end of each line. From PuTTYgen, select **Conversions -> Export OpenSSH key** to generate a text file with this format. A Linux server stores each of its private keys in /etc/ssh, such as /etc/ssh/ssh_host_rsa_key and /etc/ssh/ssh_host_dsa_key; these are text files with the correct format.

Then, as above, use show ssh-host-key to show the public key in the new pair, and distribute the public key to your SSH clients.

For example, this command sequence installs an RSA host key and then displays the public key, to be distributed to SSH-client machines:

```
bstnA(cfg)# ssh-host-key rsa
Host Key:
```

This example is a single paste operation. The CLI repeats the "Host Key" prompt after each <Return> character in the private key:

```
Host Key: -----BEGIN RSA PRIVATE KEY-----
Host Key: MIICWgIBAAKBgQC2SZyBkHgOwnB9Stp+MUMOC0cshEyCsLAovBWFUDgGisjRroBO
Host Key: 5XYm+tmMBVmHzQayILI5XwCUl31dD1O0MKj09G9GFtSs376wVG8HJmfg2dpQmZWc
Host Key: /TVBxga61c75y38aeXYWDgpOj4NU9V/jM6nz1Sy37g4EThrZjJolM9digwIBJQKB
Host Key: gB2PZXzLUb00SZfG8v+09h3z/bQxISnzI3w6LQDck3auuMgOdavt2874PvQciw8a
Host Key: VBzitR4PZ+C+nrUlFH4VuohP7vYItNY/JC95lTpLaY3sqrg8n1GmaYtVtzvtihPm
Host Key: 5Dq+a1hZIn9NPe3WsGghqFDzpubPzyx6uqpBwdYr+ppdAkEA7pYN47KnHIAxNDHT
Host Key: 9jvFXPWQj8HFduPIALQzz23MvNiB4r9QpvglQ83zJf7KK6plik2JKeD72/PED9wa
Host Key: m0wj7wJBAMOXoFLG3JM0r7J6S7R2hZ7H+pP9YAQb899olTg4XP7R2QB2JJa9bZoM
Host Key: 0c4hQY3G8GxZ9MKvdif9UOc9Yd6shq0CQQChNPSgxNHHJi8VbcaRngHk3UYARKgE
Host Key: OQqYsRwWiHWUWu/6EpBw0S3viye/uf45LfGUzJryyHLS4voYjczJy6mTAkBkcHTs
Host Key: PJrIIflbppWUBYLi1WT/4vn0RbSVUWFiDxr/ZNc3lp0qz/oC/6oCERPTLs581D9q
Host Key: 5Hrx8MiuGJoKj/IhAkAmF6NwzXu7UkWu+ToTh/RUfTpkA4JrF5yhZB5LubcdxYS5
Host Key: NilixpMbFWmXx5JFDBb2cKjLyTXKLIQ9ot9Bp3SN
Host Key: -----END RSA PRIVATE KEY-----
bstnA(cfg)# show ssh-host-key rsa

ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAo7sH/VQNMPoT67wunXKmdSGylER9/9FZtNx861Sv/xP/p5l/9hQOXNIVlxlu/6
MKUeLSTPAmmIZ1Pol0Y0dOM9iucqXTKi8aMo2088m9ylHmPwWzaATBPYkVikTm4bWhOan7EGeFa2MmQdL9ZyYHIC8L
rEfFaxo3XOWb+c3ikYk=
bstnA(cfg)# ...
```

Specifying The Windows Domains Authorized For a Group

You can associate a user group with one or more Windows domains to control that user group's access. Do this using the gbl-group mode CLI command windows-domain. This command enables you to specify the correspondence between a user account group and a Windows domain, expressed as a fully-qualified domain name.

This is part of the typical user group configuration. Refer to *Configuring Management Access*, on page 8-1 for more information for configuring user groups.

The command syntax in gbl-group mode is:

windows-domain *domainname*

where *domainname* is the fully-qualified domain name (FQDN) of the relevant Windows domain.

For example:

```
windows-domain wwmedarch.org
```

A

CLI Security Levels

Security Table

This appendix lists all CLI commands (by mode) and the administrative Security Roles that can run them. Security Roles were discussed in *Setting the Group Role*, on page 2-12. The CLI commands are listed alphabetically under each mode.

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
exec (top-level) Mode						
enable		X	X	X	X	X
save profile	X		X	X	X	X
priv-exec (enable) Mode						
activate						X
active-directory update					X	X
arp gratuitous			X	X	X	X
auto-diagnostics test					X	X
boot system			X	X	X	X
cancel			X	X	X	X
cancel migrate-metadata		X	X	X	X	X
cancel migrate-volume					X	X
cancel migration					X	X
cancel nsck report			X	X	X	X
cancel restore data		X	X	X	X	X
cancel snapshot archive		X			X	X
cancel sync			X	X	X	X
capture merge			X	X	X	X
capture session			X	X	X	X
cifs access-based-enum					X	X
cifs promote-subshares					X	X
cifs rekey			X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
clear active-license						X
clear arp			X	X	X	X
clear at						X
clear counters channel			X	X	X	X
clear counters gigabit			X	X	X	X
clear counters lacp			X	X	X	X
clear counters mgmt			X	X	X	X
clear counters redundancy			X	X	X	X
clear counters redundancy network			X	X	X	X
clear counters ten-gigabit			X	X	X	X
clear dynamic-dns				X	X	X
clear file-history archive		X			X	X
clear global-config			X	X	X	X
clear health			X	X	X	X
clear metalog usage			X	X	X	X
clear nlm locks			X	X	X	X
clear nsck			X	X	X	X
clear nvr			X	X	X	X
clear restore data		X	X	X	X	X
clear session			X	X		X
clear smtp queue			X	X	X	X
clear statistics			X	X	X	X
clear statistics api			X	X	X	X
clear statistics authentication			X	X		X
clear statistics cifs authentication			X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
clear statistics cifs path-cache			X	X	X	X
clear statistics cifs symlinks			X	X	X	X
clear statistics cifs work-queues			X	X	X	X
clear statistics domain-controller			X	X	X	X
clear statistics domain-controller load-balancing			X	X	X	X
clear statistics filer					X	X
clear statistics metadata			X	X	X	X
clear statistics metalog			X	X	X	X
clear statistics migration			X	X	X	X
clear statistics notification					X	X
clear statistics snapshot					X	X
clear subshare-cache					X	X
clear sync			X	X	X	X
clock set			X	X	X	X
close cifs file			X	X	X	X
collect			X	X	X	X
config				X		X
copy			X	X	X	X
copy global-config			X	X	X	X
copy reports		X	X	X	X	X
copy running-config			X	X	X	X
copy startup-config			X	X	X	X
delete			X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
delete startup-config			X	X	X	X
drop cifs-service user-session			X	X	X	X
drop filer-connections			X	X	X	X
dual-reboot				X	X	X
dynamic-dns update			X	X	X	X
expect			X	X	X	X
export-mapping		X	X	X	X	X
ext-filer-ip-addrs activate			X	X	X	X
firmware upgrade			X	X	X	X
global					X	X
gui				X		X
license activate						X
license activate file						X
license create license-dossier						X
load						X
move			X	X	X	X
move reports		X	X	X	X	X
nis update			X	X	X	X
nsck					X	X
password			X	X	X	X
ping license-server						X
policy					X	X
probe delegate-to					X	X
probe exports					X	X
probe metalog latency					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
raid offline			X	X	X	X
raid rebuild			X	X	X	X
raid silence			X	X	X	X
raid verify			X	X	X	X
rconsole			X	X	X	X
redundancy force-active			X	X		X
reload			X	X	X	X
remove cluster-config					X	X
remove namespace					X	X
remove service					X	X
remove-share migrate					X	X
remove-share nomigrate					X	X
remove-share offline					X	X
rename			X	X	X	X
restart					X	X
restore data		X	X	X	X	X
restore startup-config			X	X	X	X
ron evict			X	X	X	X
run			X	X	X	X
save boot-config			X	X	X	X
save profile			X	X	X	X
show subshare-cache			X	X	X	X
shutdown			X	X	X	X
smtp retry			X	X	X	X
smtp test email-event			X	X	X	X
smtp test message			X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
smtp test server			X	X	X	X
smtp welcome			X	X	X	X
snapshot		X			X	X
snapshot clear		X			X	X
snapshot manage		X			X	X
ssh-host-key generate						X
sync			X	X	X	X
sync cifs delegation			X	X	X	X
sync subshares from-namespace					X	X
sync subshares from-service					X	X
terminal beta			X	X	X	X
truncate-report			X	X	X	X
cfg Mode						
arp				X		X
at						X
channel				X		X
clock timezone				X		X
email-event				X		X
email-severity				X		X
hostname				X		X
interface gigabit				X		X
interface mgmt				X		X
interface ron				X		X
interface ten-gigabit				X		X
interface vlan				X		X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
ip domain-list				X		X
ip ftp-user				X		X
ip name-server				X		X
ip private subnet reassign				X		X
ip private vlan				X		X
ip proxy-address				X		X
ip ron-user				X		X
ip route				X		X
ip scp-user				X		X
logging destination				X		X
logging fastpath component				X		X
logging fastpath processor				X		X
logging level				X		X
login-banner				X		X
management						X
management source				X		X
monitor				X		X
nsm						X
ntp server				X		X
raid rebuild-rate				X		X
raid verification-mode				X		X
raid verification-mode automatic				X		X
raid verification-rate				X		X
redundancy				X		X
resource-profile legacy				X		X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
smtp				X		X
snmp-server community				X		X
snmp-server contact				X		X
snmp-server host				X		X
snmp-server location				X		X
snmp-server name				X		X
snmp-server traps				X		X
snmp-server traps private				X		X
snmp-server trusthost				X		X
spanning-tree				X		X
ssh-host-key						X
ssh-v1 enable						X
ssl				X		X
switch-forwarding enable				X		X
vlan				X		X
cfg-channel Mode						
description				X		X
lacp active				X		X
lacp passive				X		X
lacp rate				X		X
load-balance				X		X
members				X		X
priority				X		X
redundancy protocol				X		X
shutdown				X		X
spanning-tree shutdown				X		X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
trap shutdown				X		X
vlan				X		X
vlan-tag				X		X
cfg-email-event Mode						
description				X		X
enable				X		X
end				X		X
exit				X		X
group				X		X
group chassis event				X		X
group cifs event				X		X
group metadata event				X		X
group network event				X		X
group nsck event				X		X
group policy event				X		X
group redundancy event				X		X
group snapshot event				X		X
group stats-monitor event				X		X
group storage event				X		X
group virtual-server event				X		X
mail-to				X		X
cfg-if-gig Mode						
description				X		X
flowcontrol				X		X
redundancy protocol				X		X
shutdown				X		X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
spanning-tree				X		X
spanning-tree force-migration				X		X
spanning-tree shutdown				X		X
speed				X		X
cfg-mgmt Mode						
description				X		X
ip address				X		X
shutdown				X		X
speed				X		X
cfg-ron Mode						
heartbeat failure				X		X
heartbeat interval				X		X
ip address				X		X
peer address				X		X
shutdown				X		X
cfg-if-ten-gig Mode						
description				X		X
flowcontrol				X		X
redundancy protocol				X		X
shutdown				X		X
spanning-tree shutdown				X		X
cfg-if-vlan Mode						
description				X		X
ip address				X		X
redundancy				X		X
ron tunnel				X		X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
shutdown				X		X
cfg-if-vlan-ron-tnl Mode						
heartbeat failure				X		X
heartbeat interval				X		X
peer address				X		X
shutdown				X		X
cfg-mgmt-access Mode						
authentication						X
permit						X
cfg-redundancy Mode						
critical route				X		X
enable				X		X
peer				X		X
quorum-disk				X		X
resilver-timeout				X		X
suspend-failover				X		X
cfg-smtp Mode						
end				X		X
exit				X		X
from				X		X
mail-server				X		X
maximum age				X		X
retry interval				X		X
to				X		X
cfg-stp Mode						
forward-delay				X		X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
hello-time				X		X
mac-address aging-time				X		X
max-age				X		X
priority				X		X
protocol				X		X
shutdown				X		X
cfg-ssl Mode						
cipher				X		X
ssl-key-store				X		X
cfg-vlan Mode						
blocked-vlan				X		X
description				X		X
jumbo				X		X
members				X		X
tag				X		X
gbl Mode						
active-directory forest-trust					X	X
active-directory-forest					X	X
auto-diagnostics					X	X
cifs					X	X
cluster-name					X	X
config-replication					X	X
external-filer					X	X
file-history archive					X	X
global server					X	X
group						X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
kerberos auto-realm-traversal					X	X
kerberos health-check threshold					X	X
max-volume-groups					X	X
namespace					X	X
nfs					X	X
nfs tcp timeout					X	X
nfs-access-list						X
nis domain					X	X
ntlm-auth-db						X
ntlm-auth-server						X
policy-age-fileset					X	X
policy-cifs-attributes-fileset					X	X
policy-filename-fileset					X	X
policy-filesize-fileset					X	X
policy-intersection-fileset					X	X
policy-union-fileset					X	X
proxy-user						X
radius-server						X
schedule					X	X
user						X
windows-mgmt-auth						X
gbl-forest Mode						
child-domain					X	X
forest-root					X	X
name-server					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
tree-domain					X	X
gbl-auto-diag Mode						
additional-command					X	X
mail-to					X	X
schedule					X	X
gbl-cifs Mode						
browsing					X	X
description					X	X
domain-join					X	X
dynamic-dns					X	X
enable					X	X
export					X	X
export offline-access					X	X
kerberos-creds					X	X
signatures					X	X
wins-name-encoding					X	X
gbl-cfg-repl Mode						
description					X	X
enable					X	X
report					X	X
schedule					X	X
target-cluster					X	X
target-file					X	X
user					X	X
gbl-filer Mode						
cifs connection-limit					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
cifs-port					X	X
description					X	X
filer-type					X	X
filer-type windows					X	X
ignore-name					X	X
ip address					X	X
manage snapshots					X	X
nfs tcp connections					X	X
proxy-user					X	X
spn					X	X
gbl-archive Mode						
description					X	X
location					X	X
gbl-gs Mode						
active-directory proxy-user					X	X
description					X	X
enable					X	X
virtual server					X	X
windows-domain					X	X
gbl-gs-vs Mode						
active-directory alias					X	X
enable					X	X
wins					X	X
wins-alias					X	X
wins-name					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
gbl-gs-vs Mode						
active-directory alias					X	X
enable					X	X
wins					X	X
wins-alias					X	X
wins-name					X	X
gbl-group Mode						
role						X
user						X
windows-domain						X
gbl-ns Mode						
character-encoding nfs					X	X
cifs anonymous-access					X	X
cifs authentication kerberos					X	X
cifs authentication ntlm					X	X
cifs authentication ntlmv2					X	X
cifs filer-signatures					X	X
description					X	X
enable					X	X
enable shares					X	X
metadata cache-size					X	X
metadata share					X	X
ntlm-auth-db					X	X
ntlm-auth-server					X	X
policy					X	X
policy freespace					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
protocol					X	X
proxy-user					X	X
sam-reference					X	X
volume					X	X
windows-mgmt-auth					X	X
gbl-ns-vol Mode						
age-fileset					X	X
auto reserve files					X	X
auto sync files					X	X
cifs access-based-enum					X	X
cifs case-sensitive					X	X
cifs deny-symlinks					X	X
cifs file-system-name					X	X
cifs notify-change-mode					X	X
cifs oplocks-disable					X	X
cifs path-cache					X	X
compressed-files					X	X
description					X	X
direct					X	X
enable					X	X
enable shares					X	X
filename-fileset					X	X
filer-subshares					X	X
filesize-fileset					X	X
freespace					X	X
freespace cifs-quota					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
intersection-fileset					X	X
metadata					X	X
metadata share					X	X
modify					X	X
named-streams					X	X
nfs-param					X	X
notification rule					X	X
persistent-acls					X	X
place-rule					X	X
policy freespace					X	X
policy migrate-method					X	X
policy order-rule					X	X
policy pause					X	X
reimport-modify					X	X
reserve files					X	X
shadow					X	X
shadow-copy-rule					X	X
share					X	X
share-farm					X	X
show sid-translation					X	X
snapshot consistency					X	X
snapshot directory cifs-name					X	X
snapshot directory nfs-name					X	X
snapshot privileged-access					X	X
snapshot replica-snap-rule					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
snapshot rule					X	X
snapshot vss-mode					X	X
sparse-files					X	X
unicode-on-disk					X	X
union-fileset					X	X
volume-group					X	X
gbl-fs-age, gbl-fs-age Mode						
every					X	X
last					X	X
select-files					X	X
start					X	X
gbl-fs-name, gbl-ns-vol-fs-name Mode						
name					X	X
path					X	X
recurse					X	X
gbl-fs-filesize, gbl-ns-vol-fs-filesize Mode						
select-files					X	X
gbl-fs-isect, gbl-ns-vol-fs-isect Mode						
from					X	X
gbl-ns-vol-ntfy Mode						
enable					X	X
report					X	X
retain					X	X
schedule					X	X
gbl-ns-vol-plc Mode						
enable					X	X
from					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
inline notify					X	X
inline report					X	X
limit-migrate					X	X
migrate close-file					X	X
migrate hard-links					X	X
report					X	X
schedule					X	X
source					X	X
target					X	X
tentative					X	X
volume-scan					X	X
gbl-ns-vol-shdwcp Mode						
bandwidth-limit					X	X
cifs-8dot3-resolution					X	X
database-location					X	X
delta-threshold					X	X
enable					X	X
from					X	X
inline-notify					X	X
prune-target					X	X
publish					X	X
report					X	X
retry					X	X
schedule					X	X
sid-translation					X	X
target					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
gbl-ns-vol-shr Mode						
attach					X	X
cifs access-based-enum exclude					X	X
critical					X	X
description					X	X
enable					X	X
end					X	X
exit					X	X
filer					X	X
freespace adjust					X	X
freespace apparent-size					X	X
freespace ignore					X	X
ignore-sid-errors					X	X
import priority					X	X
import rename-directories					X	X
import rename-files					X	X
import skip-managed-check					X	X
import sync-attributes					X	X
managed-volume					X	X
migrate					X	X
policy freespace					X	X
replica-snap					X	X
sid-translation					X	X
strict-attribute-consistency					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
gbl-ns-vol-sfarm Mode						
balance					X	X
constrain-directories					X	X
constrain-files					X	X
enable					X	X
policy freespace					X	X
share					X	X
gbl-ns-vol-replica-snap Mode						
enable					X	X
exclude					X	X
report					X	X
retain					X	X
schedule					X	X
gbl-ns-vol-snap Mode						
archive					X	X
contents					X	X
enable					X	X
exclude					X	X
report					X	X
retain					X	X
schedule					X	X
gbl-fs-union, gbl-ns-vol-fs-union Mode						
from					X	X
gbl-nfs Mode						
description					X	X
enable					X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
export					X	X
nlm enable					X	X
offline-behavior					X	X
gbl-nfs-acl Mode						
anonymous-gid						X
anonymous-uid						X
deny						X
description						X
end						X
exit						X
nis domain						X
permit						X
permit netgroup						X
gbl-nis-dom Mode						
ip address					X	X
gbl-ntlm-auth-db Mode						
end						X
exit						X
gbl-ntlm-auth-srv Mode						
end						X
exit						X
ip address						X
password						X
port						X
windows-domain						X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
gbl-fs-cifs-attr Mode						
offline					X	X
gbl-proxy-user Mode						
description						X
windows-domain						X
gbl-radius Mode						
auth-port						X
key						X
retries						X
timeout						X
gbl-schedule Mode						
description					X	X
duration					X	X
every					X	X
start					X	X
stop					X	X
gbl-user Mode						
group						X
password						X
ssh-key						X
gbl-mgmt-auth Mode						
permit						X
user						X
show (any) Mode						
end	X		X	X	X	X
exit	X	X	X	X	X	X
find	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
grep	X		X	X	X	X
pause	X		X	X	X	X
ping	X		X	X	X	X
remark	X		X	X	X	X
show	X		X	X	X	X
show active-directory	X		X	X	X	X
show active-directory status	X		X	X	X	X
show active-license	X		X	X	X	X
show arp	X		X	X	X	X
show at	X		X	X	X	X
show auto-diagnostics	X		X	X	X	X
show baudrate	X		X	X	X	X
show boot	X		X	X	X	X
show capture	X		X	X	X	X
show capture sessions	X		X	X	X	X
show channel	X		X	X	X	X
show channel load-balance	X		X	X	X	X
show channel summary	X		X	X	X	X
show chassis	X		X	X	X	X
show chassis software	X		X	X	X	X
show cifs-service	X		X	X	X	X
show cifs-service client-activity	X		X	X	X	X
show cifs-service exports	X		X	X	X	X
show cifs-service kerberos-tickets	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
show cifs-service open-files	X		X	X	X	X
show cifs-service path-cache	X		X	X	X	X
show cifs-service subshares	X		X	X	X	X
show cifs-service transactions	X		X	X	X	X
show cifs-service user-sessions	X		X	X	X	X
show clock	X		X	X	X	X
show cluster	X		X	X	X	X
show config-replication	X		X	X	X	X
show configs	X		X	X	X	X
show cores	X		X	X	X	X
show documentation	X		X	X	X	X
show dynamic-dns	X		X	X	X	X
show email-event	X		X	X	X	X
show email-event all	X		X	X	X	X
show email-severity	X		X	X	X	X
show exports	X		X	X	X	X
show external-filer	X		X	X	X	X
show fastpath cifs-signatures	X		X	X	X	X
show fastpath logging	X		X	X	X	X
show fastpath resources	X		X	X	X	X
show file-history	X		X	X	X	X
show file-history archive		X			X	X
show file-history virtual-service	X	X	X	X	X	X
show filer connections	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
show filer connections ip-addr	X		X	X	X	X
show firmware upgrade			X	X	X	X
show global server	X		X	X	X	X
show global service	X		X	X	X	X
show global-config	X		X	X	X	X
show global-config archive	X		X	X	X	X
show global-config cifs	X		X	X	X	X
show global-config config-replication	X		X	X	X	X
show global-config namespace	X		X	X	X	X
show global-config nfs	X		X	X	X	X
show group all						X
show group roles						X
show group users						X
show health	X		X	X	X	X
show health time-skew	X		X	X	X	X
show history	X		X	X	X	X
show hostname	X		X	X	X	X
show id-mappings	X		X	X	X	X
show interface	X		X	X	X	X
show interface gigabit	X		X	X	X	X
show interface mgmt	X		X	X	X	X
show interface ten-gigabit	X		X	X	X	X
show interface vlan	X		X	X	X	X
show ip address	X		X	X	X	X
show ip domain	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
show ip proxy-addresses	X		X	X	X	X
show ip route	X		X	X	X	X
show license	X		X	X	X	X
show license-dossier	X		X	X	X	X
show load-balancing	X		X	X	X	X
show logging destination	X		X	X	X	X
show logging levels	X		X	X	X	X
show logs	X		X	X	X	X
show mac-address-table	X		X	X	X	X
show mac-address-table summary	X		X	X	X	X
show management	X		X	X	X	X
show master-key						X
show memory usage	X		X	X	X	X
show metalog usage	X		X	X	X	X
show monitor	X		X	X	X	X
show namespace	X		X	X	X	X
show namespace mapping					X	X
show namespace status	X		X	X	X	X
show nfs tcp	X		X	X	X	X
show nfs-access-list	X		X	X	X	X
show nfs-service	X		X	X	X	X
show nfs-service mounts	X		X	X	X	X
show nis domain	X		X	X	X	X
show nis netgroup	X		X	X	X	X
show nlm	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
show notification		X				X
show nsck	X		X	X	X	X
show nsm						X
show nsm warm-restart history	X		X	X	X	X
show ntlm-auth-db	X		X	X	X	X
show ntlm-auth-server	X		X	X	X	X
show ntp servers	X		X	X	X	X
show ntp status	X		X	X	X	X
show policy	X		X	X	X	X
show policy files-closed	X		X	X	X	X
show policy filesets	X		X	X	X	X
show policy history	X		X	X	X	X
show policy queue	X		X	X	X	X
show policy schedule	X		X	X	X	X
show processors	X		X	X	X	X
show processors usage	X		X	X	X	X
show proxy-user	X		X	X	X	X
show radius-server	X		X	X	X	X
show redundancy	X		X	X	X	X
show redundancy all	X		X	X	X	X
show redundancy ballots	X		X	X	X	X
show redundancy critical-services	X		X	X	X	X
show redundancy history	X		X	X	X	X
show redundancy license	X		X	X	X	X
show redundancy metalog	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
show redundancy network	X		X	X	X	X
show redundancy peer	X		X	X	X	X
show redundancy quorum-disk	X		X	X	X	X
show redundancy reboot-history	X		X	X	X	X
show redundancy resilver-timeout	X		X	X	X	X
show releases	X		X	X	X	X
show replicated-configs	X		X	X	X	X
show reports	X	X	X	X	X	X
show reports status	X		X	X	X	X
show reports type	X	X	X	X	X	X
show restore data	X	X	X	X	X	X
show ron	X		X	X	X	X
show ron conflicts	X		X	X	X	X
show ron database	X		X	X	X	X
show ron route	X		X	X	X	X
show ron tunnel	X		X	X	X	X
show ron tunnel all	X		X	X	X	X
show running-config	X		X	X	X	X
show schedule	X		X	X	X	X
show scripts	X		X	X	X	X
show server-mapping	X		X	X	X	X
show server-mapping status	X		X	X	X	X
show sessions			X	X		X
show shadow	X		X	X	X	X
show share status	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
show smtp queue	X		X	X	X	X
show smtp status	X		X	X	X	X
show smtp welcome	X		X	X	X	X
show snapshots	X	X	X	X	X	X
show snmp-server	X		X	X	X	X
show software	X		X	X	X	X
show spanning-tree detailed	X		X	X	X	X
show spanning-tree interface	X		X	X	X	X
show spanning-tree summary	X		X	X	X	X
show ssh-host-key						X
show ssh-user	X		X	X	X	X
show statistics api	X		X	X	X	X
show statistics authentication			X	X		X
show statistics cifs authentication	X		X	X	X	X
show statistics cifs fastpath	X		X	X	X	X
show statistics cifs path-cache	X		X	X	X	X
show statistics cifs symlinks		X	X	X	X	X
show statistics cifs work-queues	X		X	X	X	X
show statistics domain-controller	X		X	X	X	X
show statistics domain-controller load-balancing	X		X	X	X	X
show statistics filer	X		X	X	X	X
show statistics global server	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
show statistics metadata	X		X	X	X	X
show statistics metalog	X		X	X	X	X
show statistics migration	X		X	X	X	X
show statistics namespace	X		X	X	X	X
show stats-logs	X		X	X	X	X
show stats-monitor	X		X	X	X	X
show sync	X		X	X	X	X
show system tasks	X		X	X	X	X
show terminal	X		X	X	X	X
show uptime	X		X	X	X	X
show users	X		X	X	X	X
show version	X		X	X	X	X
show virtual path-history		X			X	X
show virtual service	X		X	X	X	X
show vlan	X		X	X	X	X
show vlan summary	X		X	X	X	X
show volume-group	X		X	X	X	X
show windows-mgmt-auth	X		X	X	X	X
tail	X		X	X	X	X
terminal character-set	X		X	X	X	X
terminal clear	X		X	X	X	X
terminal confirmation	X		X	X	X	X
terminal expand-prompt	X		X	X	X	X
terminal expert	X		X	X	X	X
terminal history	X		X	X	X	X
terminal length	X		X	X	X	X

CLI Command	Operator	Backup Operator	Network Tech.	Network Engineer	Storage Engineer	Crypto Officer
terminal logging	X	X	X	X	X	X
terminal stop-on-error	X		X	X	X	X
terminal timeout	X		X	X	X	X
terminal width	X		X	X	X	X
wait-for ip-routes			X	X	X	X
wait-for migration	X		X	X	X	X
wait-for nsck			X	X	X	X
wait-for remove			X	X	X	X
wait-for restore data		X	X	X	X	X
wait-for shares-online			X	X	X	X
wait-for snapshot		X			X	X
wait-for sync			X	X	X	X
wait-for vip-disable			X	X	X	X
wait-for vip-enable			X	X	X	X
wait-for volume-disable			X	X	X	X
wait-for volume-enable			X	X	X	X

Index

T

U

V

W