CHAPTER 56

Managing Software and Configurations

This chapter describes how to manage the ASASM software and configurations and includes the following sections:

Managing the Flash File System

This section includes the following topics:

Viewing Files in Flash Memory

You can view files in flash memory and see information about files as follows:

- To view files in flash memory, enter the following command:

  ```
  hostname# dir [disk0: | disk1:]
  ```

  Enter **disk0:** for the internal flash memory. The **disk1:** keyword represents the external flash memory. The internal flash memory is the default.

  For example:

  ```
  hostname# dir

  Directory of disk0:/
  500    -rw-  4958208     22:56:20 Nov 29 2004  cdisk.bin
  ```

```
2513   -rw-  4634        19:32:48 Sep 17 2004  first-backup
2788   -rw-  21601       20:51:46 Nov 23 2004  backup.cfg
2927   -rw-  8670632     20:42:48 Dec 08 2004  asdmfile.bin
```

- To view extended information about a specific file, enter the following command:

  ```
  hostname# show file information [path:/]filename
  ```

  The default path is the root directory of the internal flash memory (disk0:/).

  For example:

  ```
  hostname# show file information cdisk.bin

  disk0:/cdisk.bin:
    type is image (XXX) []
    file size is 4976640 bytes version 7.0(1)
  ```

  The file size listed is for example only.

Deleting Files from Flash Memory

You can remove files from flash memory that you no longer need. To delete a file from flash memory, enter the following command:

```
hostname# delete disk0: filename
```

By default, the file is deleted from the current working directory if you do not specify a path. You may use wildcards when deleting files. You are prompted with the filename to delete, and then you must confirm the deletion.

Downloading Software or Configuration Files to Flash Memory

You can download application images, ASDM images, configuration files, and other files to the internal flash memory or, for the ASASM, to the external flash memory from a TFTP, FTP, SMB, HTTP, or HTTPS server.

Note You cannot have two files with the same name but with different letter case in the same directory in flash memory. For example, if you attempt to download the file, Config.cfg, to a location that contains the file, config.cfg, you receive the following error message:

```
%Error opening disk0:/Config.cfg (File exists).
```

This section includes the following topics:

Downloading a File to a Specific Location

This section describes how to download the application image, ASDM software, a configuration file, or any other file that needs to be downloaded to flash memory. To download a file to the running or startup configuration, see the "Downloading a File to the Startup or Running Configuration" section on page 56-3.

For information about installing the Cisco SSL VPN client, see the *Cisco AnyConnect VPN Client Administrator Guide*. For information about installing Cisco Secure Desktop on the ASASM, see the *Cisco Secure Desktop Configuration Guide for Cisco ASA 5500 Series Administrators*.

To configure the ASASM to use a specific application image or ASDM image if you have more than one installed, or have installed them in external flash memory, see the "Configuring the Application Image and ASDM Image to Boot" section on page 56-4.

To configure the ASASM to use a specific configuration as the startup configuration, see the "Configuring the File to Boot as the Startup Configuration" section on page 56-5.

For multiple context mode, you must be in the system execution space.

To download a file to flash memory, see the following commands for each download server type:

- To copy from a TFTP server, enter the following command:

  ```
  hostname# copy tftp://server[/path]/filename {disk0:/ | disk1:/}[path/]filename
  ```

- To copy from an FTP server, enter the following command:

  ```
  hostname# copy ftp://[user[:password]@]server[/path]/filename {disk0:/ |
  disk1:/}[path/]filename
  ```

- To copy from an HTTP or HTTPS server, enter the following command:

  ```
  hostname# copy http[s]://[user[:password]@]server[:port][/path]/filename {disk0:/ |
  disk1:/}[path/]filename
  ```

- To copy from an SMB server, enter the following command:

  ```
  hostname# copy smb://[user[:password]@]server[/path]/filename {disk0:/ |
  disk1:/}[path/]filename
  ```

- To use secure copy, first enable secure shell (SSH), and then enter the following command:

  ```
  hostname# ssh scopy enable
  ```

  From a Linux client, enter the following command:

  ```
  scp -v -pw password filename username@asa_address
  ```

  The **-v** is for verbose, and if **-pw** is not specified, you will be prompted for a password.

Downloading a File to the Startup or Running Configuration

You can download a text file to the running or startup configuration from a TFTP, FTP, SMB, or HTTP(S) server, or from the flash memory.

To copy a file to the startup configuration or running configuration, enter one of the following commands for the appropriate download server:

Note When you copy a configuration to the running configuration, you merge the two configurations. A merge adds any new commands from the new configuration to the running configuration. If the configurations are the same, no changes occur. If commands conflict or if commands affect the running of the context, then the effect of the merge depends on the command. You might get errors, or you might have unexpected results.

- To copy from a TFTP server, enter the following command:

  ```
  hostname# copy tftp://server[/path]/filename {startup-config | running-config}
  ```

- To copy from an FTP server, enter the following command:

  ```
  hostname# copy ftp://[user[:password]@]server[/path]/filename {startup-config |
  running-config}
  ```

- To copy from an HTTP or HTTPS server, enter the following command:

  ```
  hostname# copy http[s]://[user[:password]@]server[:port][/path]/filename
  {startup-config | running-config}
  ```

- To copy from an SMB server, enter the following command:

  ```
  hostname# copy smb://[user[:password]@]server[/path]/filename {startup-config |
  running-config}
  ```

- To copy from flash memory, enter the following command:

  ```
  hostname# copy {disk0:/ | disk1:/}[path/]filename {startup-config | running-config}
  ```

For example, to copy the configuration from a TFTP server, enter the following command:

```
hostname# copy tftp://209.165.200.226/configs/startup.cfg startup-config
```

To copy the configuration from an FTP server, enter the following command:

```
hostname# copy ftp://admin:letmein@209.165.200.227/configs/startup.cfg startup-config
```

To copy the configuration from an HTTP server, enter the following command:

```
hostname# copy http://209.165.200.228/configs/startup.cfg startup-config
```

Configuring the Application Image and ASDM Image to Boot

By default, the ASASM boots the first application image that it finds in internal flash memory. It also boots the first ASDM image it finds in internal flash memory, or if one does not exist in this location, then in external flash memory. If you have more than one image, you should specify the image that you want to boot. For the ASDM image, if you do not specify the image to boot, even if you have only one image installed, then the ASASM inserts the **asdm image** command into the running configuration. To avoid problems with Auto Update (if configured), and to avoid the image search at each startup, you should specify the ASDM image that you want to boot in the startup configuration.

To configure the application image to boot, enter the following command:

```
hostname(config)# boot system url
```

where *url* can be one of the following:

- {**disk0:/** | **disk1:/**}[*path*/]*filename*

- **tftp://**[*user*[**:***password*]@]*server*[:*port*]**/**[*path/*]*filename*

Note The TFTP option is only supported for the ASASM.

You can enter up to four **boot system** command entries to specify different images to boot from in order; the ASASM boots the first image it finds. Only one **boot system tftp** command can be configured, and it must be the first one configured.

Note If the ASASM is stuck in a cycle of constant booting, you can reboot the ASASM into ROMMON mode. For more information about the ROMMON mode, see the "Using the ROM Monitor to Load a Software Image" section on page 57-11.

To configure the ASDM image to boot, enter the following command:

```
hostname(config)# asdm image {disk0:/ | disk1:/}[path/]filename
```

Configuring the File to Boot as the Startup Configuration

By default, the ASASM boots from a startup configuration that is a hidden file. You can alternatively set any configuration to be the startup configuration by entering the following command:

```
hostname(config)# boot config {disk0:/ | disk1:/}[path/]filename
```

Deleting Files from a USB Drive on the ASA 5500-X Series

When you delete a file from a USB drive (accessed as disk1:, for example), then the USB is moved to the other slot (from bottom to top, or top to bottom), and the file reappears. With this type of online insertion removal, to make sure that the file is actually deleted and no longer appears when you enter the **show disk1:** command, enter the following command:

```
hostname# eject disk1:
```

Performing Zero Downtime Upgrades for Failover Pairs

The two units in a failover configuration should have the same major (first number) and minor (second number) software version. However, you do not need to maintain version parity on the units during the upgrade process; you can have different versions on the software running on each unit and still maintain failover support. To ensure long-term compatibility and stability, we recommend upgrading both units to the same version as soon as possible.

Table 56-1 shows the supported scenarios for performing zero-downtime upgrades on a failover pair.

Table 56-1 Zero-Downtime Upgrade Support

Type of Upgrade	Support
Maintenance Release	You can upgrade from any maintenance release to any other maintenance release within a minor release. For example, you can upgrade from 7.0(1) to 7.0(4) without first installing the maintenance releases in between.
Minor Release	You can upgrade from a minor release to the next minor release. You cannot skip a minor release. For example, you can upgrade from 7.0(1) to 7.1(1). Upgrading from 7.0(1) directly to 7.2(1) is not supported for zero-downtime upgrades; you must first upgrade to 7.1(1).
Major Release	You can upgrade from the last minor release of the previous version to the next major release. For example, you can upgrade from 7.2(1) to 8.0(1), assuming that 7.2(1) is the last minor version in the 7.x release series. **Note** Zero downtime upgrades are possible, even when feature configuration is migrated, for example, from 8.2.x to 8.3.x.

For more details about upgrading the software on a failover pair, see the following topics:

- Upgrading an Active/Standby Failover Configuration, page 56-6
- Upgrading an Active/Active Failover Configuration, page 56-7

Upgrading an Active/Standby Failover Configuration

To upgrade two units in an Active/Standby failover configuration, perform the following steps:

Step 1 Download the new software to both units, and specify the new image to load with the **boot system** command (see the "Configuring the Application Image and ASDM Image to Boot" section on page 56-4).

Step 2 Reload the standby unit to boot the new image by entering the following command on the active unit:

```
active# failover reload-standby
```

Step 3 When the standby unit has finished reloading, and is in the Standby Ready state, force the active unit to fail over to the standby unit by entering the following command on the active unit.

Note Use the **show failover** command to verify that the standby unit is in the Standby Ready state.

```
active# no failover active
```

Step 4 Reload the former active unit (now the new standby unit) by entering the following command:

```
newstandby# reload
```

Step 5 When the new standby unit has finished reloading and is in the Standby Ready state, return the original active unit to active status by entering the following command:

```
newstandby# failover active
```

Upgrading an Active/Active Failover Configuration

To upgrade two units in an Active/Active failover configuration, perform the following steps:

Step 1 Download the new software to both units, and specify the new image to load with the **boot system** command (see the "Configuring the Application Image and ASDM Image to Boot" section on page 56-4).

Step 2 Make both failover groups active on the primary unit by entering the following command in the system execution space of the primary unit:

```
primary# failover active
```

Step 3 Reload the secondary unit to boot the new image by entering the following command in the system execution space of the primary unit:

```
primary# failover reload-standby
```

Step 4 When the secondary unit has finished reloading, and both failover groups are in the Standby Ready state on that unit, make both failover groups active on the secondary unit by using the following command in the system execution space of the primary unit:

Note Use the **show failover** command to verify that both failover groups are in the Standby Ready state on the secondary unit.

```
primary# no failover active
```

Step 5 Make sure that both failover groups are in the Standby Ready state on the primary unit, and then reload the primary unit using the following command:

```
primary# reload
```

Step 6 If the failover groups are configured with the **preempt** command, they automatically become active on their designated unit after the preempt delay has passed. If the failover groups are not configured with the **preempt** command, you can return them to active status on their designated units using the **failover active group** command.

Backing Up Configuration Files or Other Files

This section includes the following topics:

Backing up the Single Mode Configuration or Multiple Mode System Configuration

In single context mode or from the system configuration in multiple mode, you can copy the startup configuration or running configuration to an external server or to the local flash memory as follows:

- To copy to a TFTP server, enter the following command:

```
hostname# copy {startup-config | running-config} tftp://server[/path]/filename
```

- To copy to a FTP server, enter the following command:

```
hostname# copy {startup-config | running-config}
ftp://[user[:password]@]server[/path]/filename
```

- To copy to local flash memory, enter the following command:

```
hostname# copy {startup-config | running-config} {flash:/ | disk0:/ |
disk1:/}[path/]filename
```

Note Be sure that the destination directory exists. If it does not exist, first create the directory using the **mkdir** command.

Backing Up a Context Configuration or Other File in Flash Memory

Copy context configurations or other files that are on the local flash memory by entering one of the following commands in the system execution space:

- To copy to a TFTP server, enter the following command:

```
hostname# copy disk{0 | 1}:/[path/]filename tftp://server[/path]/filename
```

- To copy to a FTP server, enter the following command:

```
hostname# copy disk{0 | 1}:/[path/]filename
ftp://[user[:password]@]server[/path]/filename
```

- To copy to an SMB file-system, enter the following command:

```
hostname# copy disk{0 | 1}:/[path/]filename
smb://[user[:password]@]server[/path]/filename
```

- To copy from the ASASM using HTTPS, enter the following URL in your browser:

  https://*ASA_IP***/disk**{**0** | **1**}**/***filename*

- To copy to local flash memory, enter the following command:

  hostname# **copy disk**{**0** | **1**}**:/**[*path*/]*filename* **disk**{**0** | **1**}**:/**[*path*/]new*filename*

Note Be sure that the destination directory exists. If it does not exist, first create the directory using the **mkdir** command.

Backing Up a Context Configuration within a Context

In multiple context mode, from within a context, you can perform the following backups:

- To copy the running configuration to the startup configuration server (connected to the admin context), enter the following command:

  hostname/contexta# **copy running-config startup-config**

- To copy the running configuration to a TFTP server connected to the context network, enter the following command:

  hostname/contexta# **copy running-config tftp:/***server*[*/path*]*/filename*

Copying the Configuration from the Terminal Display

To print the configuration to the terminal, enter the following command:

hostname# **show running-config**

Copy the output from this command, and then paste the configuration into a text file.

Backing Up Additional Files Using the Export and Import Commands

Additional files essential to your configuration might include the following:

- Files that you import using the **import webvpn** command. Currently, these files include customizations, URL lists, web content, plug-ins, and language translations.
- DAP policies (dap.xml).
- CSD configurations (data.xml).
- Digital keys and certificates.
- Local CA user database and certificate status files.

The CLI lets you back up and restore individual elements of your configuration using the **export** and **import** commands.

To back up these files, for example, those files that you imported with the **import webvpn** command or certificates, perform the following steps:

Step 1 Run the applicable **show** command(s) as follows:

```
hostname # show import webvpn plug-in
ica
rdp
ssh, telnet
vnc
```

Step 2 Run the **export** command for the file that you want to back up (in this example, the rdp file):

```
hostname # export webvpn plug-in protocol rdp tftp://tftpserver/backupfilename
```

Using a Script to Back Up and Restore Files

You can use a script to back up and restore the configuration files on your ASASM, including all extensions that you import via the **import webvpn** CLI, the CSD configuration XML files, and the DAP configuration XML file. For security reasons, we do not recommend that you perform automated backups of digital keys and certificates or the local CA key.

This section provides instructions for doing so and includes a sample script that you can use as is or modify as your environment requires. The sample script is specific to a Linux system. To use it for a Microsoft Windows system, you need to modify it using the logic of the sample.

Note The existing CLI lets you back up and restore individual files using the **copy**, **export**, and **import** commands. It does not, however, have a facility that lets you back up all ASASM configuration files in one operation. Running the script facilitates the use of multiple CLIs.

This section includes the following topics:

Prerequisites

To use a script to back up and restore an ASASM configuration, first perform the following tasks:

- Install Perl with an Expect module.
- Install an SSH client that can reach the ASASM.
- Install a TFTP server to send files from the ASASM to the backup site.

Another option is to use a commercially available tool. You can put the logic of this script into such a tool.

Running the Script

To run a backup-and-restore script, perform the following steps:

Step 1 Download or cut-and-paste the script file to any location on your system.

Step 2 At the command line, enter **Perl** *scriptname*, where *scriptname* is the name of the script file.

Step 3 Press **Enter**.

Step 4 The system prompts you for values for each option. Alternatively, you can enter values for the options when you enter the **Perl** *scriptname* command before you press **Enter**. Either way, the script requires that you enter a value for each option.

Step 5 The script starts running, printing out the commands that it issues, which provides you with a record of the CLIs. You can use these CLIs for a later restore, which is particularly useful if you want to restore only one or two files.

Sample Script

```
#!/usr/bin/perl
#Function: Backup/restore configuration/extensions to/from a TFTP server.
#Description: The objective of this script is to show how to back up
configurations/extensions before the backup/restore command is developed.
# It currently backs up the running configuration, all extensions imported via "import
webvpn" command, the CSD configuration XML file, and the DAP configuration XML file.
#Requirements: Perl with Expect, SSH to the ASA, and a TFTP server.
#Usage: backupasa -option option_value
#       -h: ASA hostname or IP address
#       -u: User name to log in via SSH
#       -w: Password to log in via SSH
#       -e: The Enable password on the security appliance
#       -p: Global configuration mode prompt
#       -s: Host name or IP address of the TFTP server to store the configurations
#       -r: Restore with an argument that specifies the file name. This file is produced
during backup.
#If you don't enter an option, the script will prompt for it prior to backup.
#
#Make sure that you can SSH to the ASA.

use Expect;
use Getopt::Std;

#global variables
%options=();
$restore = 0; #does backup by default
$restore_file = '';
$asa = '';
$storage = '';
$user = '';
$password = '';
$enable = '';
$prompt = '';
$date = `date +%F`;
chop($date);
my $exp = new Expect();

getopts("h:u:p:w:e:s:r:",\%options);
do process_options();

do login($exp);
do enable($exp);
if ($restore) {
   do restore($exp,$restore_file);
}
else {
   $restore_file = "$prompt-restore-$date.cli";
   open(OUT,">$restore_file") or die "Can't open $restore_file\n";
   do running_config($exp);
   do lang_trans($exp);
```

```
   do customization($exp);
   do plugin($exp);
   do url_list($exp);
   do webcontent($exp);
   do dap($exp);
   do csd($exp);
   close(OUT);
}
do finish($exp);

sub enable {
   $obj = shift;
   $obj->send("enable\n");
   unless ($obj->expect(15, 'Password:')) {
      print "timed out waiting for Password:\n";
   }
   $obj->send("$enable\n");
   unless ($obj->expect(15, "$prompt#")) {
      print "timed out waiting for $prompt#\n";
   }
}

sub lang_trans {
   $obj = shift;
   $obj->clear_accum();
   $obj->send("show import webvpn translation-table\n");
   $obj->expect(15, "$prompt#" );
   $output = $obj->before();
   @items = split(/\n+/, $output);

   for (@items) {
     s/^\s+//;
     s/\s+$//;
     next if /show import/ or /Translation Tables/;
     next unless (/^.+\s+.+$/);
     ($lang, $transtable) = split(/\s+/,$_);
     $cli = "export webvpn translation-table $transtable language $lang
$storage/$prompt-$date-$transtable-$lang.po";
     $ocli = $cli;
     $ocli =~ s/^export/import/;
     print "$cli\n";
     print OUT "$ocli\n";
     $obj->send("$cli\n");
     $obj->expect(15, "$prompt#" );
   }
}

sub running_config {
  $obj = shift;
  $obj->clear_accum();
  $cli ="copy /noconfirm running-config $storage/$prompt-$date.cfg";
  print "$cli\n";
  $obj->send("$cli\n");
  $obj->expect(15, "$prompt#" );
}


sub customization {
  $obj = shift;
  $obj->clear_accum();
  $obj->send("show import webvpn customization\n");
  $obj->expect(15, "$prompt#" );
  $output = $obj->before();
  @items = split(/\n+/, $output);
```

```
  for (@items) {
    chop;
    next if /^Template/ or /show import/ or /^\s*$/;
    $cli = "export webvpn customization $_ $storage/$prompt-$date-cust-$_.xml";
    $ocli = $cli;
    $ocli =~ s/^export/import/;
    print "$cli\n";
    print OUT "$ocli\n";
    $obj->send("$cli\n");
    $obj->expect(15, "$prompt#" );
  }
}

sub plugin {
   $obj = shift;
   $obj->clear_accum();
   $obj->send("show import webvpn plug-in\n");
   $obj->expect(15, "$prompt#" );
   $output = $obj->before();
   @items = split(/\n+/, $output);

   for (@items) {
     chop;
     next if /^Template/ or /show import/ or /^\s*$/;
     $cli = "export webvpn plug-in protocol $_ $storage/$prompt-$date-plugin-$_.jar";
     $ocli = $cli;
     $ocli =~ s/^export/import/;
     print "$cli\n";
     print OUT "$ocli\n";
     $obj->send("$cli\n");
     $obj->expect(15, "$prompt#" );
   }
}

sub url_list {
   $obj = shift;
   $obj->clear_accum();
   $obj->send("show import webvpn url-list\n");
   $obj->expect(15, "$prompt#" );
   $output = $obj->before();
   @items = split(/\n+/, $output);

   for (@items) {
     chop;
     next if /^Template/ or /show import/ or /^\s*$/ or /No bookmarks/;
     $cli="export webvpn url-list $_ $storage/$prompt-$date-urllist-$_.xml";
     $ocli = $cli;
     $ocli =~ s/^export/import/;
     print "$cli\n";
     print OUT "$ocli\n";
     $obj->send("$cli\n");
     $obj->expect(15, "$prompt#" );
   }
}

sub dap {
   $obj = shift;
   $obj->clear_accum();
   $obj->send("dir dap.xml\n");
   $obj->expect(15, "$prompt#" );

   $output = $obj->before();
   return 0 if($output =~ /Error/);
```

```
   $cli="copy /noconfirm dap.xml $storage/$prompt-$date-dap.xml";
   $ocli="copy /noconfirm $storage/$prompt-$date-dap.xml disk0:/dap.xml";
   print "$cli\n";
   print OUT "$ocli\n";
   $obj->send("$cli\n");
   $obj->expect(15, "$prompt#" );
}

sub csd {
   $obj = shift;
   $obj->clear_accum();
   $obj->send("dir sdesktop\n");
   $obj->expect(15, "$prompt#" );

   $output = $obj->before();
   return 0 if($output =~ /Error/);

   $cli="copy /noconfirm sdesktop/data.xml $storage/$prompt-$date-data.xml";
   $ocli="copy /noconfirm $storage/$prompt-$date-data.xml disk0:/sdesktop/data.xml";
   print "$cli\n";
   print OUT "$ocli\n";
   $obj->send("$cli\n");
   $obj->expect(15, "$prompt#" );
}

sub webcontent {
   $obj = shift;
   $obj->clear_accum();
   $obj->send("show import webvpn webcontent\n");
   $obj->expect(15, "$prompt#" );
   $output = $obj->before();
   @items = split(/\n+/, $output);

   for (@items) {
     s/^\s+//;
     s/\s+$//;
     next if /show import/ or /No custom/;
     next unless (/^.+\s+.+$/);
     ($url, $type) = split(/\s+/,$_);
     $turl = $url;
     $turl =~ s/\/\+//;
     $turl =~ s/\+\//-/;
     $cli = "export webvpn webcontent $url $storage/$prompt-$date-$turl";
     $ocli = $cli;
     $ocli =~ s/^export/import/;
     print "$cli\n";
     print OUT "$ocli\n";
     $obj->send("$cli\n");
     $obj->expect(15, "$prompt#" );
   }
}

sub login {
    $obj = shift;
    $obj->raw_pty(1);
    $obj->log_stdout(0); #turn off console logging.
    $obj->spawn("/usr/bin/ssh $user\@$asa") or die "can't spawn ssh\n";
    unless ($obj->expect(15, "password:" )) {
        die "timeout waiting for password:\n";
    }

    $obj->send("$password\n");
```

```
    unless ($obj->expect(15, "$prompt>" )) {
        die "timeout waiting for $prompt>\n";
    }
}

sub finish {
    $obj = shift;
    $obj->hard_close();
    print "\n\n";

}

sub restore {
   $obj = shift;
   my $file = shift;
   my $output;
   open(IN,"$file") or die "can't open $file\n";
   while (<IN>) {
      $obj->send("$_");
      $obj->expect(15, "$prompt#" );
      $output = $obj->before();
      print "$output\n";
   }
   close(IN);
}

sub process_options {
  if (defined($options{s})) {
      $tstr= $options{s};
      $storage = "tftp://$tstr";
  }
  else {
      print "Enter TFTP host name or IP address:";
      chop($tstr=<>);
      $storage = "tftp://$tstr";
  }
  if (defined($options{h})) {
      $asa = $options{h};
  }
  else {
      print "Enter ASA host name or IP address:";
      chop($asa=<>);
  }

  if (defined ($options{u})) {
      $user= $options{u};
  }
  else {
      print "Enter user name:";
      chop($user=<>);
  }

  if (defined ($options{w})) {
      $password= $options{w};
  }
  else {
      print "Enter password:";
      chop($password=<>);
  }
  if (defined ($options{p})) {
      $prompt= $options{p};
  }
  else {
      print "Enter ASA prompt:";
```

```
        chop($prompt=<>);
    }
    if (defined ($options{e})) {
        $enable = $options{e};
    }
    else {
        print "Enter enable password:";
        chop($enable=<>);
    }

    if (defined ($options{r})) {
       $restore = 1;
       $restore_file = $options{r};
    }
}
```

Configuring Auto Update Support

Auto Update is a protocol specification that allows an Auto Update Server to download configurations and software images to many ASASMs and can provide basic monitoring of the ASASMs from a central location.

The ASASM can be configured as either a client or a server. As an Auto Update client, it periodically polls the Auto Update Server for updates to software images and configuration files. As an Auto Update Server, it issues updates for ASASMs configured as Auto Update clients.

Note Auto Update is supported in single context mode only.

This section includes the following topics:

- Configuring Communication with an Auto Update Server, page 56-16
- Configuring Client Updates as an Auto Update Server, page 56-18
- Viewing Auto Update Status, page 56-19

Configuring Communication with an Auto Update Server

To configure the ASASM as an Auto Update client, perform the following steps:

Step 1 To specify the URL of the Auto Update Server, enter the following command:

```
hostname(config)# auto-update server url [source interface] [verify-certificate]
```

where *url* has the following syntax:

```
http[s]://[user:password@]server_ip[:port]/pathname
```

SSL is used when **https** is specified. The *user* and *password* arguments of the URL are used for basic authentication when logging in to the server. If you use the **write terminal**, **show configuration** or **show tech-support** commands to view the configuration, the user and password are replaced with '********'.

The default port is 80 for HTTP and 443 for HTTPS.

The **source** *interface* keyword and argument specify which interface to use when sending requests to the Auto Update Server. If you specify the same interface specified by the **management-access** command, the Auto Update requests travel over the same IPsec VPN tunnel used for management access.

The **verify-certificate** keyword verifies the certificate returned by the Auto Update Server.

Step 2 (Optional) To identify the device ID to send when communicating with the Auto Update Server, enter the following command:

```
hostname(config)# auto-update device-id {hardware-serial | hostname | ipaddress [if-name]
| mac-address [if-name] | string text}
```

The identifier used is determined by specifying one of the following parameters:

- The *hardware-serial* argument specifies the ASASM serial number.
- The *hostname* argument specifies the ASASM hostname.
- The **ipaddress** keyword specifies the IP address of the specified interface. If the interface name is not specified, it uses the IP address of the interface used to communicate with the Auto Update Server.
- The **mac-address** keyword specifies the MAC address of the specified interface. If the interface name is not specified, it uses the MAC address of the interface used to communicate with the Auto Update Server.
- The **string** keyword specifies the specified text identifier, which cannot include white space or the characters ‘, “, , >, & and ?.

Step 3 (Optional) To specify how often to poll the Auto Update Server for configuration or image updates, enter the following command:

```
hostname(config)# auto-update poll-period poll-period [retry-count [retry-period]]
```

The *poll-period* argument specifies how often (in minutes) to check for an update. The default is 720 minutes (12 hours).

The *retry-count* argument specifies how many times to try reconnecting to the server if the first attempt fails. The default is zero.

The *retry-period* argument specifies how long to wait (in minutes) between retries. The default is five minutes.

Step 4 (Optional) To schedule a specific time for the ASASM to poll the Auto Update Server, enter the following command:

```
hostname(config)# auto-update poll-at days-of-the-week time [randomize minutes]
[retry_count [retry_period]]
```

The *days-of-the-week* argument is any single day or combination of days: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday. Other possible values are daily (Monday through Sunday), weekdays (Monday through Friday), and weekends (Saturday and Sunday).

The *time* argument specifies the time in the format HH:MM at which to start the poll. For example, 8:00 is 8:00 a.m. and 20:00 is 8:00 p.m.

The **randomize** *minutes* keyword and argument specify the period to randomize the poll time following the specified start time. The range is from 1 to 1439 minutes.

The *retry_count* argument specifies how many times to try reconnecting to the Auto Update Server if the first attempt fails. The default is zero.

The *retry_period* argument specifies how long to wait between connection attempts. The default is five minutes. The range is from 1 to 35791 minutes.

Step 5 (Optional) If the Auto Update Server has not been contacted for a certain period of time, entering the following command causes it to stop passing traffic:

```
hostname(config)# auto-update timeout period
```

The *period* argument specifies the timeout period in minutes between 1 and 35791. The default is to never time out (zero minutes). To restore the default, enter the **no** form of this command.

Use the **auto-update timeout** command to be sure that the ASASM has the most recent image and configuration. This condition is reported with system log message 201008.

In the following example, an ASASM is configured to poll an Auto Update Server with the IP address 209.165.200.224, at port number 1742, from the outside interface, with certificate verification.

The ASASM is also configured to use the hostname as the device ID and to poll an Auto Update Server every Friday and Saturday night at a random time between 10:00 p.m. and 11:00 p.m. On a failed polling attempt, the ASASM will try to reconnect to the Auto Update Server ten times, and will wait three minutes between attempts at reconnecting, as shown in the following example:

```
hostname(config)# auto-update server
https://jcrichton:farscape@209.165.200.224:1742/management source outside
verify-certificate
hostname (config)# auto-update device-id hostname
hostname (config)# auto-update poll-at Friday Saturday 22:00 randomize 60 2 10
```

Configuring Client Updates as an Auto Update Server

Entering the **client-update** command enables updates for ASASMs configured as Auto Update clients and lets you specify the type of software component (ASDM or boot image), the type or family of ASASM, revision numbers to which the update applies, and a URL or IP address from which to obtain the update.

To configure the ASASM as an Auto Update Server, perform the following steps:

Step 1 To enable client update, enter the following command:

```
hostname(config)# client-update enable
```

Step 2 Configure the following parameters for the **client-update** command that you want to apply to the ASASMs:

client-update {**component** {**asdm** | **image**} | **device-id** *dev_string* | **family** *family_name* | **type** *type*} **url** *url-string* **rev-nums** *rev-nums*}

The **component** {**asdm** | **image**} parameter specifies the software component, either ASDM or the boot image of the ASASM.

The **device-id** *dev_string* parameter specifies a unique string that the Auto Update client uses to identify itself. The maximum length is 63 characters.

The **family** *family_name* parameter specifies the family name that the Auto Update client uses to identify itself. It can be asa, pix, or a text string with a maximum length of seven characters.

The **rev-nums** *rev-nums* parameter specifies the software or firmware images for this client. Enter up to four, in any order, separated by commas.

The **type** *type* parameter specifies the type of clients to notify of a client update. Because this command is also used to update Windows clients, the list of clients includes several Windows operating systems. The ASASMs in the list may include the following:

- asa5505: Cisco 5505 ASASM
- asa5510: Cisco 5510 ASASM
- asa5520: Cisco 5520 ASASM
- asa5540: Cisco 5540 ASASM

The **url** *url-string* parameter specifies the URL for the software/firmware image. This URL must point to a file appropriate for this client. For all Auto Update clients, you must use the protocol "http://" or "https://" as the prefix for the URL.

Configure the parameters for the client update that you want to apply to all ASASMs of a particular type. That is, specify the type of ASASM and the URL or IP address from which to get the updated image. In addition, you must specify a revision number. If the revision number of the remote ASASM matches one of the specified revision numbers, there is no need to update the client, and the update is ignored.

To configure a client update for Cisco 5520 ASASMs, enter the following command:

```
hostname(config)# client-update type asa5520 component asdm url
http://192.168.1.114/aus/asdm601.bin rev-nums 8.0(1)
```

Viewing Auto Update Status

To view the Auto Update status, enter the following command:

```
hostname(config)# show auto-update
```

The following is sample output from the **show auto-update** command:

```
hostname(config)# show auto-update

Server: https://********@209.165.200.224:1742/management.cgi?1276
Certificate will be verified
Poll period: 720 minutes, retry count: 2, retry period: 5 minutes
Timeout: none
Device ID: host name [corporate]
Next poll in 4.93 minutes
Last poll: 11:36:46 PST Tue Nov 13 2004
Last PDM update: 23:36:46 PST Tue Nov 12 2004
```

Downgrading Your Software

When you upgrade to Version 8.3, your configuration is migrated. The old configuration is automatically stored in flash memory. For example, when you upgrade from Version 8.2(1) to 8.3(1), the old 8.2(1) configuration is stored in flash memory in a file called 8_2_1_0_startup_cfg.sav.

Note You must manually restore the old configuration before downgrading.

This section describes how to downgrade and includes the following topics:

Information About Activation Key Compatibility

Your activation key remains compatible if you upgrade to the latest version from any previous version. However, you might have issues if you want to maintain downgrade capability:

- Downgrading to Version 8.1 or earlier versions—After you upgrade, if you activate additional feature licenses that were introduced *before 8.2*, the activation key continues to be compatible with earlier versions if you downgrade. However if you activate feature licenses that were introduced in Version 8.2 or later versions, the activation key is not backward compatible. If you have an incompatible license key, see the following guidelines:
  - If you previously entered an activation key in an earlier version, the ASASM uses that key (without any of the new licenses you activated in Version 8.2 or later versions).
  - If you have a new system and do not have an earlier activation key, you need to request a new activation key compatible with the earlier version.
- Downgrading to Version 8.2 or earlier versions—Version 8.3 introduced more robust time-based key usage as well as failover license changes:
  - If you have more than one time-based activation key active, when you downgrade, only the most recently activated time-based key can be active. Any other keys are made inactive.
  - If you have mismatched licenses on a failover pair, downgrading will disable failover. Even if the keys are matching, the license used will no longer be a combined license.

Performing the Downgrade

To downgrade from Version 8.3, perform the following steps:

Detailed Steps

Step 1 Enter the following command:

```
hostname(config)# downgrade [/noconfirm] old_image_url old_config_url [activation-key old_key]
```

Where the **/noconfirm** option downgrades without prompting. The *image_url* is the path to the old image on disk0, disk1, tftp, ftp, or smb. The *old_config_url* is the path to the saved, premigration configuration (by default, t;his configuration was saved on disk0). If you need to revert to a pre-8.3 activation key, you can enter the old activation key.

This command is a shortcut for completing the following functions:

1. Clearing the boot image configuration (**clear configure boot**).
2. Setting the boot image to be the old image (**boot system**).
3. (Optional) Entering a new activation key (**activation-key**).
4. Saving the running configuration to startup (**write memory**). This action sets the BOOT environment variable to the old image, so when you reload, the old image is loaded.

5. Copying the old configuration to the startup configuration (**copy** *old_config_url* **startup-config**).
6. Reloading (**reload**).

For example:

```
hostname(config)# downgrade /noconfirm disk0:/asa821-k8.bin disk0:/8_2_1_0_startup_cfg.sav
```