

## Chapter 6

# Managing the File System

Introduction .....	6-2
File Naming Conventions .....	6-2
Long Filenames in Releases .....	6-3
Working with Files .....	6-4
Built-In Editor .....	6-5
Using Wildcards .....	6-5
Flash Memory .....	6-5
The Flash File System (FFS) .....	6-6
If You Clear Flash Memory Completely .....	6-7
Non-Volatile Storage (NVS) .....	6-8
Command Reference .....	6-9
activate flash compaction .....	6-9
add file .....	6-10
clear flash totally .....	6-11
clear nvs totally .....	6-11
copy .....	6-12
create file .....	6-13
delete file .....	6-14
delete nvs .....	6-15
dump nvs .....	6-16
edit .....	6-17
modify nvs .....	6-20
purge file translationtable .....	6-21
rename .....	6-22
reset file permanentredirect .....	6-23
show ffile .....	6-24
show file .....	6-26
show file permanentredirect .....	6-29
show flash .....	6-30
show flash physical .....	6-32
show nvs .....	6-33
show nvs free .....	6-34

## Introduction

The file system provides a consistent file-based interface to the physical memory devices that store data on the switch. The memory devices are flash, NVS, and card. The file system allows data, such as product software, licence information, and configuration scripts, to be stored on the switch in files with descriptive names and manipulated with a consistent set of commands, regardless of where they are physically stored. The file system provides a single directory on each storage device.

## File Naming Conventions

Files are uniquely identified by names consisting of three parts. The format for these parts is:

`[device:] filename.ext`

where:

- *device* specifies the physical memory device where the file is stored.
- *filename* is the base name, and is a descriptive name for the file. It can be from 1 to 28 characters long. Invalid characters are \* + = " | \ [ ] ; : ? / , < > . Valid characters are:
  - uppercase and lowercase letters
  - digits
  - ~ ' ! @ # \$ % ^ & ( ) \_ - { }
- *.ext* is a file name extension 1 to 3 characters long. When specified, it must be separated from *filename* by a period. Valid characters are:
  - uppercase and lowercase letters
  - digits
  - hyphen

Switches determine file types based on extensions. The following table explains some of the ones possible.

Extension	File Type/Function
acc	Accounting information
bin	Bootloader software image file
cfg	Configuration or boot script
core	Core memory dump
hlp	Help file
htm	HTML file used by the HTTP server
lic	Licence information
log	Log file
mds	Modem script
paz	Compressed patch
pkg	Product software package
rez	Compressed product release file
rnd	Random number data

Extension (cont)	File Type/Function (cont)
rsc	GUI resource file
scp	Script
sec	SNMP engine information
txt	Generic text file
lfn	In long filename translation table

- If a file name contains spaces or an equals sign, it must be in double quotes.
- If a colon is in the file name, the device parameter is ignored and the switch assumes that the file name includes the device name.

For information about wildcards, see [“Using Wildcards” on page 6-5](#).

**Example** The following are examples of valid file names:

```
config.scp
"home office.log"
```

The following is an example of an illegal file name because the forward slash is not a valid delimiter for flash, and directories are not supported:

```
flash:/sys/head_o.cfg
```

## Long Filenames in Releases

All software release files support short filenames (DOS 8.3 format). Software version 2.5.1 and later support long filenames in either DOS 16.3 or DOS 28.3 format. The following table summarises which software versions support different DOS filename formats.

Software release	Dos 8.3 format	DOS 16.3 format	DOS 28.3 format
2.4.x and earlier	Yes	No	No
2.5.1 and later	Yes	Yes	No
2.6.4 and later	Yes	Yes	Yes

### Upgrading to new versions

When upgrading to this software version from earlier software releases, filenames retain their DOS naming format. DOS 8.3 format filenames remain in DOS 8.3 format and DOS 16.3 format filenames remain in DOS 16.3 format.

### Back to previous versions

If you install a software version that supports DOS 28.3 format, and then you install a previous software version that supports **only** DOS 8.3 format, long filenames that were in DOS 28.3 format are truncated to DOS 8.3 format. When you reinstall a software version that supports DOS 28.3 format, these truncated filenames are restored to their DOS 28.3 format and no information is lost.

If you install a software version that supports DOS 28.3 format, and then you install a previous software version that supports DOS 16.3 format, long filenames in DOS 28.3 format are permanently truncated to DOS 8.3 format. For example, the file AB12345678.SCP is permanently renamed AB123~01.SCP. Any long filenames that were in DOS 28.3 format remain truncated in DOS 8.3 format when you reinstall a software version that supports DOS 28.3 format.

## Working with Files

---

The switch supports file names that are up to 28 characters long and with 3-character extensions (DOS 28.3 format). However, the switch **stores** files in the 8-character DOS 8.3 format with a 3-character extension. For example, the file named extralongfilenam.cfg can be saved as extral~1.cfg in the flash file system. Therefore, files can be accessed with two file names, either of which can be used for file management.

A translation table named longname.lfn converts file names between DOS 28.3 format and DOS 8.3 format. To reconcile file names, the switch checks this translation table, which is synchronised with file contents in memory. If the translation table is corrupted, it can be rebuilt from valid files in memory. To resynchronise the translation table to the file contents in memory, use the **purge file translationtable** command on page 6-21.

The **update** option restores valid long file names to the appropriate table entries after the table has been rebuilt. Long file names are deleted that are not reconciled to the new table. All table entries are deleted that are not confirmed to be in memory. This leaves a translation table that has maintained all of its previously valid data, and disposed of the rest. The table continues to support subsequent long file name creation and management.

The **all** option completely rebuilds the translation table. All long file names are lost. The table continues to support all subsequent long file name creation and management.

To display the contents of the translation table, which converts file names between DOS 28.3 format and DOS 8.3 format, use the command:

```
show file=longfile.lfn
```

To display a directory of the files stored on the switch in both flash and NVS, use the command:

```
show file
```

To limit the display to certain files, use the command:

```
show file=filename
```

*filename* can contain wildcard characters \* and |.

To permanently delete a file, use the command:

```
delete file=filename
```

*filename* can contain wildcard characters \* and |. Note that you cannot use this command to delete the preferred software release or the current boot configuration file.

To create a text file, use the switch's built-in editor by using the command:

```
edit [filename]
```

To load a file onto a switch with HTTP, TFTP, or ZMODEM, use the command:

```
load file=filename
```

To change the name of a file, use the command:

```
rename src_filename dest_filename
```

To make a copy of a file with a new name or on a different storage device, use the command:

```
copy src_filename dest_filename
```

## Built-In Editor

The switch has a built-in full-screen text editor for editing ASCII text files stored on the switch. You can use the editor to edit your current configuration file, or to create a script file that you can run manually or automatically.

The editor uses VT100 command sequences and should be used only with a VT100-compatible terminal, terminal emulation program, or Telnet client.

To start the editor with a new file or an existing file, enter the [edit command on page 6-17](#).

## Using Wildcards

Some file commands can process groups of files as well as single files. The asterisk ( `*` ) can be used as a wildcard character in these commands to identify a group of files.

The `>` character specifies a range of characters. For example `a>z` matches any letter in the alphabet.

The `+` character specifies a list of options. For example `x*.scp+y*.scp` would specify files that match `x*.scp` or `y*.scp`.

Square brackets specify a group of operations. For example, `ppp*.[scp+cfg]` matches scripts and configuration files whose names start with "PPP".

A vertical bar matches any single character. For example, `|||.scp` matches script files with names three characters long (excluding extension and device name).

The following are examples of valid wildcard expressions:

```
flash:*. *  
*:*.rez
```

## Flash Memory

---

Flash memory allows the switch to store large volumes of data (up to 6 or 8 MBytes, depending on the model). It can store any type of file; product software, patches, and configuration files are stored in flash by default. Product software can be loaded into flash memory from a remote server over a switch port using the Loader functionality. Multiple files can be loaded and then individually selected at runtime by the Install functionality. Comprehensive management features are provided to examine the state of flash memory and to view or modify the contents.

To enable flash memory to support applications other than software releases, it is structured like a disk subsystem with files that can be created, deleted, read, and written by any switch module. Files can also be manipulated directly using

the command line interface. This allows flash to be used to store any type of data, including releases, configurations, and logs.

Flash memory is non-volatile memory that can be erased and reprogrammed many times in situ. Flash memory has advantages over other types of non-volatile memory in that it has a very large storage capacity and does not require power from a battery to retain stored data.

A limitation of flash is that it has a fixed erase block size so that individual bytes cannot be changed without first clearing a whole block of data. Additionally, there is a limit on the number of erase cycles that can be done although the limit is quite high—typically at least 100,000 cycles. This would allow three erases per day for 100 years before the limit would be exceeded.

To display the amount of flash memory installed, use the [show system command on page 4-54 of Chapter 4, Configuring and Monitoring the System](#).

To display detailed information about flash memory, use the [show flash physical command on page 6-32](#).

See the Hardware Reference for the switch for more information about memory specifications.

## The Flash File System (FFS)

The Flash File System (FFS) provides additional functionality to what the file system provides in order to manage the peculiarities of flash technologies. The additional functionality of the FFS includes:

- header and data integrity is ensured with a checksum mechanism.
- all flash processes can recover from a power cycle without data loss.
- automatic recovery of deleted file space by the compaction process.

Information about the state of the FFS can be displayed by using the [show flash command on page 6-30](#).

### Working with FFS files

You can manage FFS files like other files on the switch by using the standard file system commands:

```
edit [filename]
delete file=filename
load=filename
show file [=filename]
```

In addition, you can use the following commands to manage files stored in flash memory. To display a directory of the files stored in flash memory, use the command:

```
show ffile [check]
```

If **check** is specified, the file data checksum is also verified. This is an option because it takes longer to complete a check on large files. A file data check is also carried out each time the system reads a file.

**Compaction** Flash memory has a granular erase structure that requires data to be erased in large blocks rather than as individual bytes. To allow files to be mapped onto this structure, the FFS keeps track of the status of each file—whether it is being written, is complete, or is deleted.

The switch automatically compacts flash memory when a maximum threshold of deleted files is reached. Compaction searches through flash memory, copying good files to a new location. After the switch has copied the good files in an erase block, it clears the block. This creates space for new files by freeing up the space that was occupied by deleted files.

When a large amount of flash memory is in use, compaction may take several minutes. However, the switch continues to operate during the compaction process. A message appears when flash compaction begins; another one appears when it finishes.



**Caution** While flash is compacting, do not restart the switch or use commands that affect the flash file subsystem such as **create**, **edit**, **load**, **rename**, or **delete**. Wait until you get a message that file compaction is complete. Interrupting flash compaction may damage files.

Compaction can be manually initiated with the command:

```
activate flash compaction
```

**FFS messages** Some FFS processes generate messages in the system log (displayed with the [show log command on page 61-36 of Chapter 61, Logging Facility](#)) which include FFS message codes. See “Flash File System Message Codes” on [page B-7 of Appendix B, Reference Tables](#) for a list of codes and their meanings.

## If You Clear Flash Memory Completely



**Caution** Do not completely clear flash memory. Files with product software, licence information, and install information are stored in flash and clearing it destroys them.

### To recover from accidentally clearing flash

1. **Boot up with default configuration.**

Reboot the switch from a terminal connected to the asynchronous terminal port (not Telnet). Use the install override to run the default configuration (see the Hardware Reference for more information).

2. **Log in.**

Log into the switch by using the default password *friend* for the *manager* account.

3. **Put current software version release on server.**

Make sure you have the current product software and patch files on a server connected to the switch. Current files can be downloaded from [www.alliedtelesis.com/support/updates.html](http://www.alliedtelesis.com/support/updates.html).

4. **Assign an IP address.**

Assign an IP address to the switch interface over which the software files are to be loaded.

#### 5. Load software files onto the switch.

Load the required software and patch onto the switch. See [“Loading Files onto the Switch” on page 5-4 of Chapter 5, Managing Configuration Files and Software Versions.](#)

#### 6. Set the install information.

Set the switch to use the software installed. See [“Install Process” on page 5-14 of Chapter 5, Managing Configuration Files and Software Versions.](#)

#### 7. Reconfigure the switch.

If you have a copy of the recent configuration file stored on your network, you can download it onto the switch too. Otherwise, you must re-enter the configuration.



**Caution** While flash is compacting, do not restart the switch or use commands that affect the flash file subsystem such as **create**, **edit**, **load**, **rename**, or **delete**. Wait until you get a message that file compaction is complete. Interrupting flash compaction may damages files.

If you accidentally restart the switch, or use commands that affect the subsystem, contact your authorised distributor or reseller. You might have to return the switch to the factory.

## Non-Volatile Storage (NVS)

Non-Volatile Storage (NVS) provides a facility to store information so that it is not destroyed when the switch is reset or powered off. The type of information that may be stored in the NVS are module configuration tables, interface configurations, patches and script files, but not the boot configuration file itself.

The NVS is organised as blocks of contiguous memory of varying size. A block ID and an index uniquely identifies each block and an owner ID indicates which module created the block. NVS blocks are normally maintained by the modules that created them, but this can also be done manually.

To display information about each block in the NVS including ID, index, owner, size, and creation date, use the [show nvs command on page 6-33.](#)

To display the amount of free space in the NVS along with the size of the largest block that can be created, use the [show nvs free command on page 6-34.](#)

To delete blocks, use the [delete nvs command on page 6-15.](#) Or delete all blocks with the [clear nvs totally command on page 6-11.](#)

To display data in the NVS blocks, use the [dump nvs command on page 6-16.](#) To change data, use the [modify nvs command on page 6-20.](#)

The switch's file subsystem provides a file-based interface to NVS memory that lets you use NVS to store scripts and other files (see [“Working with Files” on page 6-4.](#))



## Command Reference

---

This section describes the commands available on the switch to support day-to-day operational and management activities.

The shortest valid command is denoted by capital letters in the Syntax section. See “[Conventions](#)” on page lxvi of [About this Software Reference](#) for details of the conventions used to describe command syntax. See [Appendix A, Messages](#) for a complete list of messages and their meanings.

### activate flash compaction

---

**Syntax** ACTivate FLash COMPACTION

**Description** This command activates the flash compaction process so you can recover space before the preset threshold is reached that triggers an automatic compaction. The compaction process is usually automatic so manual compaction is not required during normal operations.

Compaction is the process of cleaning up garbage (deleted files) by searching through flash memory, copying valid files to a new block, and erasing old blocks. This operation deletes files in the block being cleared and frees space for new files.

Compaction is necessary because flash memory has a granular erase structure that requires data to be erased in large blocks rather than as individual bytes. To allow files to be mapped onto this structure, the FFS keeps track of the status of each file—whether it is being written, is complete, or is deleted. When the total amount of flash memory for deleted files reaches a preset limit, a compaction process begins.

Compaction may take several seconds when a large amount of flash is involved. However, flash memory operations are unaffected by the process.



**Caution** While flash is compacting, do not restart the switch or use commands that affect the flash file subsystem such as **create**, **edit**, **load**, **rename**, or **delete**. Wait until you get a message that file compaction is complete. Interrupting flash compaction may damages files.

While compaction is underway, the [show flash command on page 6-30](#) indicates an FFS global operation is compacting. When finished, the global operation shows “none”.

**Related Commands** [show flash](#)

## add file

**Syntax** `ADD File=filename [COMmand=commandstring]  
[SCRipt=scriptname] [PERManentredirect] [LIMIT=limit]`

**Description** This command takes output from a specific command or script and adds it to a text file when you next issue that command or script. This is useful for collecting debug output. If a file does not exist, one is created. While output is being redirected, the text file cannot be edited, renamed, deleted, or uploaded.

Parameter	Description
File	Name of the text file where you want to send output. One is created if it does not already exist. The <i>filename</i> is in the format <i>[device:]filename.txt</i> and can be: uppercase and lowercase letters digits # \$ % & ! ' ( ) + , - . ; = @ [ ] ^ _ ` { } ~ and space <i>device</i> indicates the physical location where the file is stored. The default is flash. Default: no default
COMmand	Command whose output is used to generate the text when it is next issued. <i>Commandstring</i> is the command syntax enclosed in quotes. <b>Command</b> and <b>script</b> are mutually exclusive.
SCRipt	Script whose output is used to generate the text when it is next issued. The script is treated as a simple list of commands. Flow control statements are <b>not</b> accepted to ensure that the extra text the script produces is not in the output file. <i>Scriptname</i> has the same format as <i>filename</i> except it must have either a .cfg or .scp extension. <b>Command</b> and <b>script</b> are mutually exclusive.
PERManentredirect	Permanently directs output to the designated text file until the <b>reset file permanentredirect</b> command is issued or the switch is rebooted.
LIMIT	A decimal number from 0 to 1048576 bytes specifying the maximum file size. Default: 204800 bytes

**Examples** To add output one time only from the **show trace** command to a file called trace.txt command, use the command:

```
add fi=trace.txt com="show trace"
```

To permanently add output from the **show debug** command to a file called debug2.txt command, use the command:

```
add fi=debug2.txt com="show debug"
```

**Related Commands** [create file](#)  
[reset file permanentredirect](#)  
[show file permanentredirect](#)

## clear flash totally

---

**Syntax** CLear FLash TOTAlly

**Description** This command completely clears the file system in flash memory by erasing and reformatting it. Clearing flash is not required for normal operations. This command is intended for troubleshooting, and requires a user with Security Officer privilege when the switch is in security mode.

Erasing flash may take several minutes. While it is underway, the [show flash command on page 6-30](#) shows that the FFS global operation is “erasing”. When the operations finishes, “Erasure is successfully completed” is displayed and the global operation shows “none”.



**Caution** This command destroys all files in flash, including essential ones with product software, licence information, and install information. Files cannot be salvaged after flash has been cleared.

**Related Commands** [delete file](#)  
[clear nvs totally](#)

## clear nvs totally

---

**Syntax** CLear NVS TOTAlly

**Description** This command completely clears the file system in non-volatile storage (NVS) memory by erasing and reformatting it. Clearing NVS is not required for normal operations. This command is intended for troubleshooting, and requires a user with Security Officer privilege when the switch is in security mode.



**Caution** This command destroys all files in NVS. You cannot salvage files after you clear NVS.

**Related Commands** [delete file](#)  
[modify nvs](#)  
[show nvs](#)  
[show nvs free](#)  
[clear flash totally](#)

## copy

---

**Syntax** `COPy [device:]src-filename.ext [device:]dest-filename.ext`

When Stacking is enabled:

```
COPy [source-hostid:][device:]filename1.ext
     [dest-hostid:][device:]filename2.ext
```

where:

- *src-filename* is the source file and name of an existing file in the format `[device:]filename.ext`. Valid characters are:
  - uppercase and lowercase letters
  - digits
  - `~'!@#$%^&()-_{ }`*.ext* is an extension, such as `.txt` or `.cfg`. The original file and the copy must have the same extensions.
- *dest-filename* is name of a destination file in the same format as *src-filename*. The filename must not already exist.
- *source-hostid* indicates the host ID of the stack member that holds the file to be copied. This variable is optional but if present, the *dest-hostid* is required; if not present, the **copy** command functions in the standard way.
- *dest-hostid* indicates stack members to receive the copied file:
  - a unique number from 1 to 32
  - a range of unique numbers from 1 to 32
  - a comma-separated list
 This variable is optional but if present, the *source-hostid* is required; if not present, the **copy** command functions in the standard way.

**Description** This command copies files to another location.

**Example** To copy the file `admin.cfg` on NVS to the file `admin2.cfg` on flash, use the command:

```
cop nvs:admin.cfg admin2.cfg
```

To copy `file1.txt` from a stack member with host ID 1 to stack members with host IDs 2 through 4, and also 6, and rename it "`file2.txt`", use the command:

```
cop 1:file1.txt 2-4,6:file2.txt
```

To copy `file1.txt` from the stack member with host ID 1 to the stack member with host ID 2 without renaming it, use the command:

```
cop 1:file.txt 2:file.txt
```

While connected to the stack member with host ID 1, use a host-directed command to direct host ID 2 to copy `file.txt` and rename it "`file2.txt`":

```
2:cop file.txt file2.txt
```

Note that you **cannot** use a host-directed command to direct the **copy** command to more than one stack member at a time. For example, "`1-3: cop file.txt file2.txt`" returns an error.

Related Commands [delete file](#)  
[rename](#)  
[show file](#)

## create file

**Syntax** `CREate File=filename [FORCE] [COMmand=commandstring]  
[SCRipt=scriptname] [PERManentredirect] [LIMIT=limit]`

**Description** This command creates a text file containing output from a specific command or script. This is useful for collecting debug output. The file cannot be edited, renamed, deleted, or uploaded while it is receiving input.

Parameter	Description
File	Name of the text file that you want to create. The <i>filename</i> is in the format [ <i>device</i> :] <i>filename</i> .txt and can be: uppercase and lowercase letters digits # \$ % & ! ' ( ) + , - . ; = @ [ ] ^ _ ` { } ~ and space <i>device</i> indicates the physical location where the file is stored. The default is flash. Default: no default
FORCE	Overwrites the text file if one already exists. If <b>force</b> is not specified and the file exists, the command has no effect.
COMmand	Command whose output is used to generate the text when it is next issued. <i>Commandstring</i> is the command syntax enclosed in quotes. <b>Command</b> and <b>script</b> are mutually exclusive.
SCRipt	Script whose output is used to generate the text when it is next issued. The script is treated as a simple list of commands. Flow control statements are <b>not</b> accepted to ensure that the extra text the script produces is not in the output file. <i>Scriptname</i> has the same format as <i>filename</i> except it must have either a .cfg or .scp extension. <b>Command</b> and <b>script</b> are mutually exclusive.
PERManentredirect	Permanently directs output to the designated text file until the <a href="#">reset file permanentredirect</a> command is issued or the switch is rebooted.
LIMIT	A decimal number from 0 to 1 048 576 bytes specifying the maximum file size. Default: 204 800 bytes

**Examples** To permanently direct all debug output from the BGP module to a file named bgp.txt, use the command:

```
cre fi=bgp.txt com="enable bgp debug=all" perm
```

Related Commands [add file](#)  
[reset file permanentredirect](#)  
[show file permanentredirect](#)

## delete file

---

**Syntax** `DELEte File=filename`

where *filename* is a file identifier in the format [*device:*]filename.ext. Valid characters are:

- uppercase and lowercase letters
- digits
- ~ ' ! @ # \$ % ^ & ( ) \_ - { }

*device* specifies the physical location where the file is stored, which is flash.

If a colon is in the file name, it is assumed that it includes a device name.

For information about wildcards, refer to [“Using Wildcards” on page 6-5](#).

**Description** This command deletes specific files. It requires a user with security officer privilege when the switch is in security mode.

The GUI resource file that the switch is currently set to use can be deleted when the GUI is disabled. GUI resource files have an .rsc extension. Use the **show install** command and check the Current Install section in the output to see which resource file is currently set. See the [disable gui command on page 3-20 of Chapter 3, Using the Graphical User Interface \(GUI\)](#) for more information about disabling the GUI.

Note that you cannot delete the preferred software release or the current boot configuration file with this command. If you want to delete the files without specifying new preferred files, first use the **delete install=pref** command or **set config=none** to stop the files from being preferred.



**Caution** Files that contain patches, product software, licences, and configurations are vital to the operation of the switch and should be deleted only after careful consideration.

**Examples** To delete all the patch files on the switch, use the command:

```
delete file=*:*.pat
```

To delete the file startup1.cfg, use the command:

```
del startup1.cfg
```

To delete all script files on all storage devices, use the command:

```
del fi=*:*.scp
```

**Related Commands** [copy](#)  
[rename](#)  
[show file](#)

## delete nvs

---

**Syntax** `DELEte NVS Block=id INDeX=index`

where:

- *id* is the block identifier in hexadecimal.
- *index* is the block index in hexadecimal.

**Description** This command deletes a block from Non-Volatile Storage (NVS), and requires a user with Security Officer privilege when the switch is in security mode. The block must be identified by **block** and **index**.

**Examples** To delete the NVS block with a block id and index of 99, use the command:

```
del nvs b=99 index=99
```

**Related Commands**

- [clear nvs totally](#)
- [dump nvs](#)
- [modify nvs](#)
- [show nvs](#)
- [show nvs free](#)

## dump nvs

**Syntax** DUMP NVS [BLOCK=*id*] [INDEX=*index*] [LENGTH=*length*]  
[OFFSET=*offset*] [SIZE={BYTE | LONG | WORD}]

where:

- *id* is the block ID in hexadecimal.
- *index* is the block index in hexadecimal.
- *length* is the length of data to be dumped in hexadecimal.
- *offset* is the offset into the data to start dumping from in hexadecimal.

**Description** This command dumps data from a non-volatile storage (NVS) block (Figure 6-1, Table 6-1). The **size** parameter specifies whether the data should be displayed grouped as bytes, longwords, or words.

The **block**, **index**, **length**, **offset**, and **size** parameters are required the first time the command is used after a reboot; thereafter, they are optional. If not specified, values are used from the previous invocation.

If **offset** is not specified, the dump continues from the end of the previous display. If **offset** is specified without a value, the value from the previous invocation is used.

Figure 6-1: Example output from the **dump nvs** command

ID: 32    Index : 03    Offset: 00000000    Length: 00000050    Size: LONG					
Offset	Data				ASCII
-----	-----				-----
00000000	00010040	00000000	00000000	00000000	...@.....
00000010	00010000	00000004	00000007	00000000	.....
00000020	00000000	00000000	00000000	00000000	.....
00000030	00000000	00000000	00000000	00000000	.....
00000040	00000000	00000000	00000000	00000000	.....
-----	-----				-----

Table 6-1: Parameters in output of the **dump nvs** command

Parameter	Meaning
ID	Block ID (in hexadecimal) of the block displayed.
Index	Block index in (hexadecimal) of the block displayed.
Offset	Offset (in hexadecimal) of the data displayed.
Length	Length of data in (hexadecimal) displayed.
Size	Whether the units of data are displayed as bytes, longwords, or words.
Offset	Offset of the current record from the ID, Index and Offset specified in the header.
Data	The data.
ASCII	ASCII representation of the data.



**Related Commands** [clear nvs totally](#)  
[delete nvs](#)  
[modify nvs](#)  
[show nvs](#)  
[show nvs free](#)

## edit

---

**Syntax** `EDit [filename]`

where *filename* is in the format [*device:*]filename.ext. Valid characters are:

- uppercase and lowercase letters
- digits
- ~ ' ! @ # \$ % ^ & ( ) \_ - { }

*device* indicates the physical location where the file is stored, either flash or NVS. The default is flash. If a colon is in the filename, it is assumed that it includes a device name.

*.ext* is a 3-letter text file extension such as .cfg and .txt.

**Description** This command invokes the switch's built-in full-screen text editor to edit a text file. This command requires a user with Security Officer privilege when the switch is in security mode.

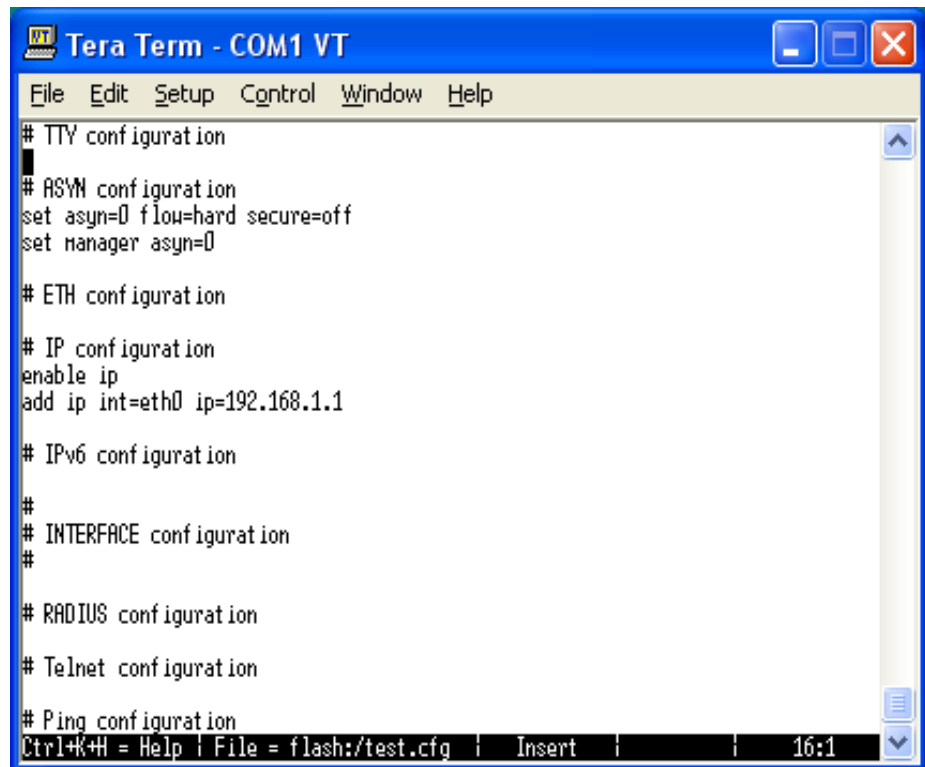
If a filename is specified and it already exists, then the editor loads it on the system. If no filename is specified, the editor prompts you for one when you exit it.

Before starting the editor make sure your terminal, terminal emulation program, or Telnet client is 100% compatible with a VT100 terminal. The editor uses VT100 control sequences to display text on the terminal, and accepts command sequences in the following table.

Cursor Movement		Delete	
or Ctrl+Z	Up one line	Ctrl+T	Delete word right
or Ctrl+X	Down one line	Ctrl+Y	Delete line
	Right one character		
	Left one character	Block Operations	
Ctrl+B	Start of file	Ctrl+K,B	Begin block mark
Ctrl+D	End of file	Ctrl+K,D	Unmark block
Ctrl+A	Start of line	Ctrl+K,U	Cut block to buffer
Ctrl+E	End of line	Ctrl+K,C	Copy block to buffer
Ctrl+U	Up one screen	Ctrl+K,V	Paste block from buffer
Ctrl+V	Down one screen	Ctrl+K,Y	Delete block
Ctrl+F	Word right		
Search		Exit	
Ctrl+K,F	Find text	Ctrl+K,X	Exit editor; save file
Ctrl+L	Repeat last find	Ctrl+C	Quit editor; do not save file
Miscellaneous			
Ctrl+I	Insert mode	Ctrl+O	Overstrike mode
Ctrl+W	Refresh the screen	Ctrl+K,H	Display help screen
Ctrl+K,O	Open a file		

Figure 6-2 shows an example of the text editor screen. The VT100 screen supports 24 lines, unlike a PC. Lines 1–23 display the text of the file being edited; line 24 at the bottom of the screen is the status bar and command line. The status bar displays the current filename, line and column position in the file, and the editing mode (overstrike or insert). When additional command information is required such as a filename or search text, a prompt is displayed in the status bar.

Figure 6-2: The editor screen layout.



The editor edits one file at a time but you can transfer text between files by using the cut and paste operations.

Obtain help at any time while in the editor by pressing [Ctrl+K,H]; that is, holding down the Ctrl key, pressing K, then the H key.

**Examples** To edit a file in flash named show.scp, use the command:

```
ed show.scp
```

To edit a file in NVS called setup1.cfg, use the command:

```
ed nvs:setup1.cfg
```

**Related Commands** [delete file](#)  
[show file](#)

## modify nvs

**Syntax** `MODify NVS Block=id INDeX=index Offset=offset  
Size={Byte|Long|Word} Value=value-list`

where:

- *id* is a block ID number in hexadecimal.
- *index* is a block index number in hexadecimal.
- *offset* is the offset in hexadecimal within the block where the values should be written.
- *value-list* is a list of values in hexadecimal, separated by commas.

<b>Description</b>	This command allows the contents of a Non Volatile Storage (NVS) block to be modified. The block must be identified by <b>block</b> and <b>index</b> .
--------------------	--

The command contiguously writes data values into the block starting at the specified **offset** and padded to **size** (length). None of the data values can require more space than the specified **size**.

**Examples** To set the third byte of the NVS block with a block ID and index of 99, to the value 99, use the command:

```
mod nvs b=99 ind=99 o=3 s=b v=fe
```

**Related Commands**

- `clear nvs totally`
- `delete nvs`
- `dump nvs`
- `show nvs`
- `show nvs free`

---

## purge file translationtable

---

**Syntax** PURge FILE Translationtable={ALl|UPdate}

**Description** This command resynchronises the translation table to file contents in memory. It is possible that the translation table and memory could become unsynchronised, for example in the event of a power outage during a file creation or deletion operation. This might result in files in memory that are not represented in the translation table, and visa versa.

The **all** option completely rebuilds the translation table. All long filenames are lost. The table continues to support subsequent long filename creation and management.

The **update** option restores all valid long filenames to the appropriate table entries after the table has been rebuilt. Long filenames that are not reconciled to the new table and table entries that are not confirmed to be in memory are deleted. This leaves a translation table that has maintained all of its previously valid data, and disposed of the rest. The table continues to support all subsequent long filename creation and management.

**Examples** To rebuild the translation table and remove all long filenames, use the command:

```
pur fi t=al
```

To rebuild the translation table and attempt to recover all long filename data, use the command:

```
pur fi t=up
```

**Related Commands** [show file](#)

## rename

**Syntax** `REName src-filename dest-filename`

**Description** This command renames files and requires a user with security officer privilege when the switch is in security mode. Please note that you cannot rename the preferred software release or the current boot configuration file.



**Caution** Files that contain product software, patches, licences, and configurations are vital to the operation of the switch and should be renamed only after careful consideration.

Parameter	Description
<i>src-filename</i>	<p>Filename of an existing source file. <b>When</b> the source is a secure file type, such as LIC or RND, its filename extension must match that of destination file.</p> <p>Default: no default</p> <p>With the following:</p> <ul style="list-style-type: none"> <li>in [device:] filename..ext format. When no device is specified, flash is assumed. Source and destination devices must be the same.</li> <li>uppercase and lowercase letters</li> <li>digits</li> <li>~ ' ! @ # \$ % ^ &amp; ( ) _ - { }</li> </ul>
<i>dest-filename</i>	<p>Filename of the destination file, which must not already exist.</p> <p>Default: no default</p> <p>With the following:</p> <ul style="list-style-type: none"> <li>in [device:] filename .ext format. When no device is specified, flash is assumed. Source and destination devices must be the same.</li> <li>uppercase and lowercase letters</li> <li>digits</li> <li>~ ' ! @ # \$ % ^ &amp; ( ) _ - { }</li> </ul>

**Examples** To rename boot.cfg to saveboot.cfg, use the command:

```
ren boot.cfg saveboot.cfg
```

**Related Commands**

- [copy](#)
- [delete file](#)
- [show file](#)

## reset file permanentredirect

---

**Syntax** RESET File[=*filename*] PERManentredirect

**Description** This command closes one or all text files so that they no longer receive input from commands or scripts. After the file closes, it can be uploaded or edited

Parameter	Description
File	<p>Name of the text file to close. If no file is specified, all text files are closed.</p> <p>The <i>filename</i> is in the format [<i>device</i>:]<i>filename.txt</i> and can be:</p> <ul style="list-style-type: none"><li>uppercase and lowercase letters</li><li>digits</li><li># \$ % &amp; ! ' ( ) + , - . ; = @ [ ] ^ _ ` { } ~ and space</li></ul> <p><i>device</i> indicates the physical location where the file is stored. The default is flash.</p> <p>Default: no default</p>

**Examples** To reset the bgp.txt file so that it no longer receives output from the **enable bgp debug=all** command (previously set), use the command:

```
reset fi=bgp.txt perm
```

**Related Commands** [add file](#)  
[create file](#)  
[show file permanentredirect](#)

## show ffile

**Syntax** `SHOW FFile[=file-identifier] [CHECK]`

where *file-identifier* is a valid FFS file identifier in the format module\filename.ext. Wildcards are allowed in any of the elements. Valid characters are:

- uppercase and lowercase letters
- digits
- ~ ' ! @ # \$ % ^ & ( ) \_ - { }

**Description** This command displays a list of the files in the Flash File System (FFS) that match the specified file identifier (Figure 6-3, Table 6-2). If a file identifier is not specified, all files are displayed. Wildcards can be used to replace any part of the file identifier to allow a more selective display.

The **check** parameter specifies that the file data checksums are to be verified. Output with this parameter may require a few more seconds for larger files.

Figure 6-3: Example output from the **show ffile** command

module	name	type	size	file date & time	address	check
	ops	cfg	2610	18-Feb-2003 03:50:12	FECD734C	-
	help	hlp	94790	21-Jan-2003 07:57:41	FECC005C	-
	config	ins	32	03-Mar-2003 10:24:43	FEB05DC0	-
	gui	ins	64	19-Feb-2003 05:41:52	FECD7EDC	-
	prefer	ins	64	28-Feb-2003 06:08:59	FEADD1B4	-
	longname	lfn	60	18-Feb-2003 03:54:54	FECD7E60	-
	feature	lic	39	21-Jan-2003 07:57:59	FECD72E4	-
	random	rnd	3904	03-Mar-2003 10:44:43	FEB05E20	-
	d_410e00	rsc	2449712	19-Feb-2003 09:09:09	FECD7F5C	-
inst	release	lic	96	18-Feb-2003 03:54:09	FECD7DC0	-
load	melistst	paz	6108	03-Mar-2003 10:24:09	FEB045A4	-
load	52-251	rez	2795756	28-Feb-2003 05:59:36	FE82F12C	-
-----						
flash use:						
	files .....	5354100 bytes (12 files)				
	garbage ....	178988 bytes				
	free .....	1675872 bytes				
	block size .	131072 bytes				
	total .....	7340032 bytes				
-----						

Table 6-2: Parameters in output of the **show ffile** command

Parameter	Meaning
module	Module that created the file.
name	filename.
type	File type.
size	Size of the file in bytes shown as a decimal number.
file date & time	Date and time the file was created.
address	Base address of the file in hexadecimal.



Table 6-2: Parameters in output of the **show ffile** command (cont)

Parameter	Meaning
check	Result of the file data check (if <b>check</b> was specified).
files	Number of bytes of flash memory used by valid files.
garbage	Number of bytes of flash memory used by deleted files.
free	Number of bytes of flash memory free.
total	Total size of flash memory.

**Examples** To display all the release files created by the Loader module, use the command:

```
sh ff=load\*.rez
```

**Related Commands** [show nvs](#)  
[show file](#)

## show file

---

**Syntax** `SHoW FiLe[=filename] [DEvice={ALl|FLash|NVs}]`

where *filename* is in the format [*device:*]filename.ext. Valid characters are:

- uppercase and lowercase letters
- digits
- ~ ' ! @ # \$ % ^ & ( ) \_ - { } \* > [ ]

Wildcard characters \* may appear in the filename when displaying them, but not when creating them. The wildcard character matches any string.

Character ranges may be specified using the > character, for example a>z matches any letter in the alphabet. The + character may be used to specify a list of options, for example a\*.scp+b\*.scp would specify files that match a\*.scp or b\*.scp.

Square brackets may be used, for example ppp\*.[scp+cfg] matches scripts and configuration files whose names start with "ppp".

The vertical bar | character matches any single character. For example, | | |.scp matches script files with names three characters long (excluding extension and device name).

**Description** This command displays a list of the files in the file subsystem that match the specified filename ([Figure 6-4 on page 6-27](#), [Table 6-3 on page 6-27](#)). Wildcards can be used to replace any part of the file identifier to allow a more selective display. If the filename specifies a single file and it exists, the contents are displayed.

This command requires a user with security officer privilege when the switch is in security mode.

The **device** parameter specifies the physical storage devices whose files are to be listed. This parameter is ignored if the file name includes a device name.

To display the contents of the translation table, which converts filenames between DOS 28.3 format and DOS 8.3 format, use the **show file=longfile.lfn** command ([Figure 6-5](#), [Table 6-4 on page 6-27](#)).

Figure 6-4: Example output from the **show file** command

Filename	Device	Size	Created	Locks
12345678901234567890.scp	flash	24	29-Mar-2004 15:34:21	0
13gggggg.scp	flash	8	29-Mar-2004 15:34:03	0
16a.scp	flash	7	17-Mar-2004 10:50:33	0
16abcd.scp	flash	14	17-Mar-2004 10:21:24	0
16ffff.scp	flash	32	16-Mar-2004 13:41:26	0
16ffffff.scp	flash	8	16-Mar-2004 14:17:19	0
409275.scp	flash	507	03-Nov-2003 12:07:37	0
409275a.scp	flash	441	24-Oct-2003 12:23:04	0
409451.scp	flash	588	10-Nov-2003 10:17:18	0
86263aka.rez	flash	3604528	16-Apr-2004 14:20:46	0
atobrsa.key	flash	321	04-Feb-2004 14:32:51	0
basic.cfg	flash	119	01-Dec-2003 15:35:56	0
bgp.cfg	flash	2811	15-Apr-2004 10:22:40	0
bgppeer.scp	flash	35	16-Apr-2004 09:59:20	0
cck.scp	flash	1018	14-Oct-2003 15:27:57	0
client.cfg	flash	2679	06-Nov-2003 13:38:48	0
config.ins	flash	32	19-Apr-2004 12:07:50	0

Table 6-3: Parameters in output of the **show file** command

Parameter	Meaning
Filename	Name of the file.
Device	Device where the file is physically stored, such as flash.
Size	Size of the file in bytes as a decimal number.
Created	Date and time the file was created.
Locks	Number of concurrent processes using the file.

Figure 6-5: Example output from the **show file=longfile.lfn** command

short filename	device	long filename	created	size	check
123456~0.scp	flash	12345678901234567890.scp	15:34:21	24	0

Table 6-4: Parameters in output of the **show file=longfile.lfn** command

Parameter	Meaning
Short filename	Name of the file in DOS 8.3 format.
Device	Device where the file is physically stored, such as flash.
Long filename	Name of the file in DOS 28.3 format.
Created	Date and time the file was created.
Size	Size of the file in bytes, as a decimal number.
Check	For flash and NVS files this value is 0 and not used.

**Examples** To display all patch files on a switch, use the command:

```
sh fi=*.paz
```

To display the contents of the config.scp script file, use the command:

```
sh fi=config.scp
```

To display the contents of the longfile.lfn long filename table, use the command:

```
sh fi=longfile.lfn
```

**Related Commands**

- [delete file](#)
- [purge file translationtable](#)
- [show flash](#)
- [show nvs](#)

## show file permanentredirect

**Syntax** `SHoW FiLe[=filename] PERManentredirect`

**Description** This command displays information about one text file or all that are permanently receiving output from commands or scripts (Figure 6-6, Table 6-5). These files are typically created to collect data during debugging.

The **file** parameter displays information about a specific text file (Figure 6-7). The *filename* option is in the format [device:]filename.txt and can be:

- uppercase and lowercase letters
- digits
- # \$ % & ! ' ( ) + , - . ; = @ [ ] ^ \_ ` { } ~ and space

*Device* indicates the physical location where the file is stored. The default is flash.

Figure 6-6: Example output from the **show file permanentredirect** command

TTY Instance	Current Size	Limit	File
17	12345	204800	bgp.txt

Figure 6-7: Example output from the **show file=*filename* permanentredirect** command

File.....	bgp.txt
TTY Instance....	17
Current Size....	12345
Limit.....	204800
Input(s).....	COMMAND="enable bgp debug=all"

Table 6-5: Parameters in output of the **show file permanentredirect** command

Parameter	Meaning
TTY Instance	Instance number for the TTY device. For details about the TTY device, see the <a href="#">show tty command on page 62-29 of Chapter 62, Terminal Server</a> .
Current Size	Size of the text file in bytes.
Limit	Limit of file size in bytes set by the <b>limit</b> parameter.
File	Name of text file.
Input(s)	Commands and scripts that generate input for the text file.

**Examples** To display all text files receiving output from commands or scripts, use the command:

```
sh fi perm
```

**Related Commands** [add file](#)  
[create file](#)  
[reset file permanentredirect](#)

## show flash

**Syntax** SHow FLash [FFs]

**Description** This command displays information about the file system stored in flash memory. The Flash File System (FFS) provides a consistent file-based interface to the physical flash memory structure, and housekeeping and management functions (Figure 6-8, Table 6-6).

Figure 6-8: Example output from the **show flash** command

```
FFS info:
global operation ..... none
compaction count ..... 256
est compaction time ... 88 seconds
files ..... 1420044 bytes (4 files)
garbage ..... 19652 bytes
free ..... 526384 bytes
required free block ... 131072 bytes
total ..... 2097152 bytes

diagnostic counters:
event      successes      failures
-----
get         0             0
open        0             1
read        0             0
close       0             0
complete    0             0
write       0             0
create      0             0
put         0             0
delete      0             0
check       0             0
erase       0             0
compact     0             0
verify      0             0
-----
```

Table 6-6: Parameters in output of the **show flash** command

Parameter	Meaning
global operation	Global operation currently running; either none, restarting, erasing, compacting, or verifying.
compaction count	Number of times the flash has been compacted since the last total erasure.
est compaction time	Estimate of how long compaction would take if it was started now.
files	Amount of space used by valid files.
garbage	Amount of space used by deleted files.
free	Amount of free space.
required free block	Minimum contiguous working space. This amount of flash memory must remain available. Therefore, it is not included in the "free" entry.
total	Total flash size.

Table 6-6: Parameters in output of the **show flash** command (cont)

Parameter	Meaning
diagnostic counters	Counts of the successes and failures for each type of FFS operation.

FFS failure counts do not necessarily mean that an error has occurred, but are also incremented if the specified file could not be found. For example, attempting to delete a file that does not exist results in the delete failures count being incremented.

**Related Commands**    [show file](#)  
                              [show nvs](#)

## show flash physical

**Syntax** SHow FLash Physical

**Description** This command displays physical information about the specific type of flash installed in the switch ([Figure 6-9](#), [Table 6-7](#)).

Figure 6-9: Example output from the **show flash physical** command

```
total size ..... 16 MBytes
  available to FFS ... 15 MBytes
  available to boot .. 1 MBytes
device type ..... 28F128
devices ..... 1
location ..... built in
programming power ..... off
block erase time ..... 1000 milliseconds
total erase blocks .... 128
  FFS erase blocks ... 120
  Boot erase blocks .. 8
erase block size ..... 128 kBytes
erase bit state ..... 1
page buffers ..... 1
size of page buffer ... 32 bytes
```

Table 6-7: Parameters in output of the **show flash physical** command

Parameter	Meaning
total size	Amount of flash memory installed.
available to FFS	Amount of flash memory available to the Flash Filing System.
available to boot	Amount of flash memory available to the boot flash.
device type	Type of flash device installed.
devices	Number of flash devices installed.
location	Whether flash memory is built in or a SIMM stick.
programming power	Whether programming power is on or off.
block erase time	Time taken to erase an erase block.
total erase blocks	Number of erase blocks.
FFS erase blocks	Number of erase blocks available to the Flash Filing System.
Boot erase blocks	Number of erase blocks available to the Boot system.
erase block size	Size of each erase block, in bytes.
erase bit state	State of an erased bit.
page buffers	Number of page buffers.
size of page buffer	Byte size of each page buffer.

**Related Commands** [show flash](#)



## show nvs

**Syntax** `SHoW NVS [BLOCK=id [INDeX=index]]`

where:

- *id* is a block ID number in hexadecimal.
- *index* is a block index number in hexadecimal.

**Description** This command displays information about the file system stored in Non-Volatile Storage (NVS) memory. It requires a user with security officer privilege when the switch is in security mode.

If the **block** parameter is specified, blocks with the specified ID are shown. If the **index** parameter is specified, the block with the specified ID and index are shown (Figure 6-10, Table 6-8).

Figure 6-10: Example output from the **show nvs** command

Block ID	Index	Size (bytes)	Creation Date	Creator ID	Block Address
0000001a	00000002	00000178	11-Aug-2000	00000012	ffe00200
0000001a	00000003	0000001a	01-Aug-2000	00000012	ffe00400
00000032	00000002	00000050	19-Feb-2001	00000022	ffe06000
00000038	00000000	00000000	01-Aug-2000	00000021	ffe00000
00000043	00000001	000000b4	27-Mar-2001	00000029	ffe02a00
00000043	00000002	00000f78	23-Mar-2001	00000029	ffe02e00
00000043	00000003	00000058	09-Mar-2001	00000029	ffe04c00
00000043	00000004	00000340	14-Nov-2000	00000029	ffe04200
00000043	00000005	00000340	14-Nov-2000	00000029	ffe04600
00000043	00000006	000004e8	20-Nov-2000	00000029	ffe06200
00000043	00000007	00000130	19-Dec-2000	00000029	ffe03000
00000043	00000008	00000088	21-Dec-2000	00000029	ffe08000
00000045	000003fc	00000de4	27-Mar-2001	00000026	ffe00a00
00000045	000003fd	00000024	27-Mar-2001	00000026	ffe00800

Table 6-8: Parameters in output of the **show nvs** command

Parameter	Meaning
Block ID	ID of the block in hexadecimal.
Index	Index of the block in hexadecimal.
Size (bytes)	Size of the block in hexadecimal bytes.
Creation Date	Date the block was created. "***-***-***" indicates that the date was undefined when the block was created.
Creator ID	ID of the module that created the block.
Block Address	Pointer to battery backed RAM where the block starts.

**Related Commands**

- [delete nvs](#)
- [dump nvs](#)
- [modify nvs](#)
- [show file](#)
- [show flash](#)
- [show nvs free](#)

## show nvs free

---

**Syntax** SHow NVS FRee

**Description** This command shows how much free space there is in the Non-Volatile Storage (NVS) and the size of the largest block that can be created ([Figure 6-11](#)).

Figure 6-11: Example output from the **show nvs free** command.

Number of free sectors	85
Number of bytes in free sectors	85656

**Related Commands**

- [clear nvs totally](#)
- [delete nvs](#)
- [dump nvs](#)
- [modify nvs](#)
- [show nvs](#)