Chapter 24

Encryption Services

Introduction

This chapter describes the data security services available on the switch, how the services are provided, the switch network functions which use these services, and how to monitor the services.

Data Encryption

Data encryption for switches is driven by the need for organisations to keep sensitive data private and secure. Encrypting network data before it is passed to the wide area network (WAN) ensures that the data can not be read or modified as it crosses the WAN. Since all wide area traffic passes through the switch, the switch is the ideal place to locate the complex hardware required to provide secure data encryption. Locating the encryption function in the network switch and integrating the complex encryption key management procedures into the switch's management system minimises the cost of supporting an encrypted network.

Data encryption operates by applying an encryption algorithm and key to the original data (the plaintext) to convert it into an encrypted form (the ciphertext). The ciphertext produced by encryption is a function of the algorithm used and the key. Since it is easy to discover what type of algorithm is being used the security of an encryption system relies on the secrecy of its key information. When the ciphertext is received by the remote router the decryption algorithm and key are used to recover the original plaintext. Often a checksum is also added to the data before encryption to allow the validity of the data to be checked on decryption.

This section describes the following types of encryption:

- **Symmetrical Encryption**
- **Asymmetrical (Public Key) Encryption**

Symmetrical Encryption

Symmetrical encryption refers to algorithms in which a single key is used for both the encryption and decryption processes. Anyone who has access to the key used to encrypt the plaintext can decrypt the ciphertext. Because the encryption key must be kept secret to protect the data these algorithms are also called private, or secret key algorithms. The key can be any value of the appropriate length.

DES

The most common symmetrical encryption system is the *Data Encryption Standard* (DES) algorithm (FIPS PUB 46). The DES algorithm has proven to be a highly secure encryption algorithm. To fully conform to the DES standard the actual data encryption operations must be carried out in hardware. Software implementations can only be DES-compatible, not DES-compliant. The DES algorithm has a key length of 56 bits and operates on 64-bit blocks of data. DES can be used in the following modes:

Mode	Description
Electronic Code Book (ECB)	The fundamental DES function. Plaintext is divided into 64-bit blocks that are encrypted with the DES algorithm and key. ECB always produces the same block of ciphertext for a given input block of plaintext.

Mode	Description
Cipher Block Chaining (CBC)	The most popular form of DES encryption. CBC operates on 64-bit blocks of data, but includes a feedback step that chains consecutive blocks so that repetitive plaintext data (such as ASCII blanks) does not yield identical ciphertext. CBC introduces a dependency between data blocks, which protects against fraudulent data insertion and replay attacks. Feedback for the first block of data is provided by a 64-bit Initialisation Vector (IV). This is the DES mode used for the switch's data encryption process.
Cipher FeedBack (CFB)	An additive stream cipher method that uses DES to generate a pseudo-random binary stream that is combined with the plaintext to produce the ciphertext. The ciphertext is then fed back to form a portion of the next DES input block.
Output FeedBack (OFB)	Combines the first IV with the plaintext to form ciphertext. The ciphertext is then used as the next IV.

The DES algorithm has been optimised to produce very high speed hardware implementations, making it ideal for networks where high throughput and low latency are essential.

Asymmetrical (Public Key) Encryption

Asymmetrical encryption algorithms use two keys—one for encryption and one for decryption. The encryption key is called the *public key* because it cannot be used to decrypt a message and therefore does not have to be kept secret. Only the decryption, or private key needs to be kept secret. The other name for this type of algorithm is *public key encryption*. The public and private key pair cannot be randomly assigned but must be generated together. In a typical scenario, a decryption station will generate a key pair and then distribute the public key to encrypting stations. This distribution need not be kept secret, but must be protected against the substitution of the public key by a malicious third party.

Another use for asymmetrical encryption is as a digital signature. The signature station publishes its public key, and then signs its messages by encrypting them with its private key. To verify the source of a message the receiver decrypts the messages with the published public key. If the message that results is valid then the signing station is authenticated as the source of the message.

The most common asymmetrical encryption algorithm is RSA. RSA utilises mathematical operations which are relatively easy to calculate in one direction but which have no known reverse solution. The security of RSA relies on the difficulty of factoring the modulus of the RSA key. Because typical key lengths of 512 bits or greater are used in public key encryption systems, decrypting RSA encrypted messages is almost impossible with current technology.

Asymmetrical encryption algorithms require enormous computational resources, making them very slow when compared to symmetrical algorithms. For this reason they are normally only used on small blocks of data (e.g. exchanging symmetrical algorithm keys), and not for entire data streams.

ENCO Services

The ENCO module provides services to user applications via channel pairs. A user application requests a service, specifying any configuration needed for the service, and is attached to an ENCO channel pair if the service and free channels are available. A channel pair consists of an encoding channel and a decoding channel. An encoding channel is used for encryption, and a decoding channel is used for decoding.

The ENCO module provides the following services:

- Encryption—DES encryption, RSA encryption, and RSA key generation
- Key storage

The number of channels available depends on the amount of RAM on the switch. Switches with up to 8 MBytes of RAM can have up to 512 encryption channels. Switches with 16 MBytes can have up to 1024 channels, and switches with 32 Mbytes up to 2048 channels. To check the amount of RAM on a switch, use the **show system** command on page 4-49 of Chapter 4, Configuring and Monitoring the System.

The identification number of the lowest and highest channels available can be displayed by using the **show enco** command on page 24-13. This command also displays general information about the ENCO module and available services.

A user module that requests the retention of process histories between packets for an encryption service may also request that the history of one of its channels be reset. Whenever a decoding channel gets out of step with its associated encoding channel the encoding channel's history must be reset.

Encryption

For management purposes, the ENCO module provides RSA and 56-bit single DES encryption to the Secure Shell and Secure Sockets Layer modules. SSL uses RSA indirectly because it uses PKI, and PKI uses RSA. For more information, see Chapter 26, Secure Shell, Chapter 27, Secure Sockets Layer (SSL), and Chapter 29, Public Key Infrastructure (PKI).

Key Creation and Storage

Keys required by the switch for encryption and authentication services are created and stored by the ENCO module. Keys are stored in the flash memory.

The following types of keys are used by the switch:

- DES, used for DES encryption
- RSA, used for RSA encryption

Each key that is created and stored in the switch has a unique identification number. Each key has a set of attributes, including the user module associated with the key. Each user module specifies whether these key attributes must be set. For example, SSH uses the IP address attribute of RSA keys to find the public key of the remote peer.

Key Formats

Keys can be stored in the switch and displayed in several formats.

DES keys can be entered in a proprietary short/checked ASCII format or in hexadecimal format. The short/checked ASCII format represents each 5 bits of the key by an ASCII character. The legal characters are lowercase letters and digits (2–9). The digits 0 and 1 are not used, to prevent confusion with the letters O and I. A checksum field is added to ensure the key has been entered correctly. DES keys are displayed in both hexadecimal and the short/checked ASCII format.

RSA keys can not be entered via the command line. They can be generated on the switch or imported from a text file. See "RSA Keys" on page 24-5 and the **create enco key** command on page 24-7 for more information about importing RSA keys.

See the **create enco key** command on page 24-7 for more information about generating and entering keys into the switch. See the **show enco key** command on page 24-29 for more information about displaying keys.

RSA Keys

RSA keys can be generated using the **create enco key** command on page 24-7. RSA public keys generated on an external device (such as a PC) can be imported as a text file. RSA public keys on the switch can be exported to a text file for transfer to an external device.

Generating RSA keys is time consuming and processor intensive. Two very large random prime numbers must be created and then combined to form the RSA key. The fastest method for generating large prime numbers is to create a large random number and then test to see if it is prime. If is not, then the number is modified and tested again. Because this is a very random procedure, it can take anywhere between 3 seconds and 30 minutes, depending on the size of the key and the CPU. To ensure the normal operation of the switch is not effected, RSA key generation is performed as a background task.

RSA public keys can be imported from and exported to text files. RSA public keys generated by an external device can be loaded on to the switch in text file format (`.key` files) and ENCO RSA public keys are generated from these text files. When an RSA key generated by the switch is exported to a text file, only the public portion of the key is written to the file.

The switch recognises three text file formats—SSH (Figure 24-1), NIQ (Figure 24-2) and hex (Figure 24-3). The default format is hex. The first number in the key file is the length of the RSA public key in bits; the second number is the exponent field of the key; the last number is the modulus field of the key. The length, exponent, and modulus fields in SSH format files are in one long line. In Figure 24-1 the backward slash (bold) shows where the line wraps. The length, exponent, and modulus fields are on separate lines in NIQ and hex formats.

Figure 24-1: An RSA public key file in SSH format

```
512 65537 582714540409421726755748030187077328862507329405933811534665149936372 69\
25543081397311308147828977987913742520391622516348731783649991255114050692755958 77
```

Figure 24-2: An RSA public key file in NIQ format

```
-NiQ Switch RSA Public Key
512
65537
582714540409421726755748030187077328862 5
073294059338115346651499363720692554308 1
397311308147828977987913742520391622516 3
487317836499912551140506927559587 7
```

Figure 24-3: An RSA public key file in HEX format

```
512
0x010001
0x6f427e5112e1389e2af1c4df09545fa88f90b093aabbdebb5778ef5ed1d39fe9
248602ef11e399216b52adae2f5fd1ae8b7ca5c19b3c27a3ec5179966cb58465
```

See the **create enco key** command on page 24-7 for more information about generating and entering keys into the switch.

Access Control

When encryption is configured and enabled, the switch must be in *security mode*. See the **enable system security_mode** command on page 23-45 of Chapter 23, User Authentication for details.

Caution Keys created on a switch not already in security mode are destroyed when the switch is restarted. Enable security mode before creating encryption keys.

When the switch is in security mode, only users with security officer privilege can execute commands that could impact the security of the switch and its keys.

A user must login from a local port, a Secure Shell session or a Telnet session from a remote security officer address (if the Remote Security Officer function is enabled) to gain security officer privilege.

See Chapter 23, User Authentication for more information about creating users with security officer privilege, configuring remote security officers, and logging into a user account with security officer privilege. See Chapter 26, Secure Shell for more information about Secure Shell.

When an encrypting switch is removed from service its sensitive encryption keys should be cleared before it is transported in an unprotected manner. To clear all encryption keys on a switch by disabling security mode, use the **disable system security_mode** command on page 23-41 of Chapter 23, User Authentication.

Important The encryption technology used in the switch is a government controlled product. The switch encryption hardware must never be disposed of by the user. These products must always be returned to your authorised distributor or reseller for deregistration and disposal.

Command Reference

This section describes the commands available on the switch to configure and monitor the encryption processes on the switch.

A user must be logged in with security officer privilege to configure encryption services. See Chapter 23, User Authentication for more information about creating users with security officer privilege, configuring remote security officers, and logging in with security officer privilege.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page xxxviii of About this Software Reference in the front of this manual for details of the conventions used to describe command syntax. See Appendix A, Messages for a complete list of messages and their meanings.

create enco key

Syntax

```
CREate ENCo KEY=key-id TYPe={DES|RSA}
   [DESCription=description] [FILe=filename] [FORmat={HEX|
   NIQ|SSH}] [IPaddress=ipadd] [LENgth=length]
   [MODule=module-id] [{RANDom|VALue=value}]
```

where:

- *key-id* is a number from 0 to 65535.
- *description* is a character string 1 to 24 characters long. Valid characters are any printable character. If *description* contains spaces, it must be in double quotes.
- *filename* is a valid switch filename with a .key extension.
- *ipadd* is an IP address in dotted decimal notation.
- *length* is a number from 0 to 65535.
- *module-id* is the name or number of a switch module (see "Module Identifiers and Names" on page B-2 of Appendix B, Reference Tables for a complete list).
- *value* is a character string of variable length depending on the key type. For ASCII formatted keys, valid characters are lowercase letters and digits (2-9). The digits 0 and 1 are illegal, to prevent confusion with the letters O and l. Hexadecimal keys must start with "0x" and must contain an even number of characters using the digits (0-9), and letters (a-f). Passphrase keys may contain any printable character. If *value* contains spaces, it must be in double quotes.

Description

This command creates a specific type of encryption key, and stores the key information in the switch's flash memory. You can also import or export RSA keys with this command. This command requires a user with security officer privilege when the switch is in security mode.

The **key** parameter specifies the identification number for the key.

The **type** parameter specifies the type of key to be created. If a DES key is being created, then the **random** or **value** parameters must be specified. If **des** is specified, a 56-bit DES key is created. If an RSA key is being generated, then the **length** or **file** parameters must be specified. If the **file** parameter is specified, the RSA key is imported from or exported to the specified file. If the **file** parameter is not specified, then a random RSA key is generated.

The **description** parameter specifies a user-defined description for the key, to make it easier to keep track of different keys.

The **file** parameter specifies name of a switch file. RSA public keys may be imported from or exported to a file in either Secure Shell format, the switch's own format or in hexadecimal format. If the file exists but the specified RSA key does not exist, then the RSA key is imported from the file. If the specified RSA key does exist but the file does not exist, the RSA key is exported to the file. The **format** parameter must be specified when importing or exporting keys.

The **format** parameter specifies the format of the key file when importing or exporting an RSA key. Secure Shell users should use SSH. NIQ is the switch's own format, which can be used for transferring RSA keys between switches. Hex format should be used when transferring keys between other devices. The default is **hex**. If **format** is specified, the **file** parameter must also be present.

The **ipaddress** parameter specifies an IP address to associate with the key. The ISAKMP and SSH modules use this value to find the RSA public key of a remote peer. Documentation for particular modules specify whether this parameter is required.

The **length** parameter specifies the length of the key to be created. An RSA key length is specified in bits and must be a multiple of 32. Valid RSA keys are from 256 to 2048 bits. A key of **general** type can have any length from 2 to 64 bytes.

The **module** parameter can be used to link a key to a specific module. Documentation for particular modules specify whether this parameter is required.

The **random** parameter creates a random key. The key can entered into another switch by using the **create enco key** command and specifying the **value** parameter.

The **value** parameter creates a key with the supplied value. DES keys require a key in 5-bit ASCII format or hexadecimal format. This ASCII representation includes a check value of the key to ensure it has been typed in correctly. A hexadecimal key always starts with "0x". The value of a key can be displayed by using the **show enco key** command.

Examples To create a random DES encryption key with the identification number 1 and then display the key, use the commands:

```
cre enc key=1 typ=des rand
sh enc key=1
```

To add this key to the other switch, use the command:

```
cre enc key=1 type=des value=value
```

on the other switch, where *value* is the value of the key displayed in the output of the **show enco key** command.

To create a random 512-bit RSA private key with the key identification number 2, use the command:

```
cre enc key=2 typ=rsa len=512 desc="Switch A private key"
```

To create an uploadable file for the public component of the same RSA key in the format used by Secure Shell use the command:

```
cre enc key=2 typ=rsa fil=routerA.key for=ssh
```

To import an RSA key from the file `RSA.key`, which is in hex format, as encryption key 3, use the command:

```
cre enc key=3 typ=rsa fil=rsa.key for=hex
```

Related Commands

destroy enco key
set enco key
show enco key

destroy enco key

Syntax

`DESTroy ENCo KEY=`*`key-id`* `LOCation=FLAsh`

where *key-id* is a number from 0 to 65535

Description

This command destroys the specified encryption key. The memory the key occupied is overwritten to ensure the key is irretrievable. This command requires a user with security officer privilege when the switch is in security mode.

The **key** parameter specifies the identification number for the key. A key with the specified identification number must exist.

The **location** parameter specifies the location of the key.

Examples

To destroy the encryption key in flash with the key identification number 4, use the command:

```
dest enc key=4 loc=fla
```

Related Commands

create enco key
set enco key
show enco key

disable enco debugging

Syntax `DISable ENCo DEBugging={CHannel|PAcket|TImestamp|ALL}`

Description This command disables debugging for the ENCO module.

The **debug** parameter specifies the debugging option to disable. Specify **packet** to disable debugging of the contents of packets processed by the ENCO module. Specify **timestamp** to disable measurements of encryption and compression operations. Specify **all** to disable all debugging options. The **channel** option is no longer supported.

Examples To disable the debugging of the contents of packets processed by the ENCO module, use the command:

```
dis enc deb=pa
```

Related Commands **enable enco debugging**
show enco debug

enable enco debugging

Syntax `ENAble ENCo DEBugging={CHannel|PAcket|TImestamp|ALL}`

Description This command enables debugging for the ENCO module. Debugging information is sent to the terminal from which the command was entered.

The **debug** parameter specifies the debugging option to enable. Specify **packet** to enable debugging of the contents of packets processed by the ENCO module. Specify **timestamp** to enable measurements of encryption and compression operations. Specify **all** to enable all debugging options. The **channel** option is no longer supported.

Examples To enable the debugging of the contents of packets processed by the ENCO module, use the command:

```
ena enc deb=pa
```

Related Commands **disable enco debuggingshow enco debug**

reset enco counter

Syntax

```
RESET ENCo COUnter={ALl|DEs|HMac|JObprocessing|QUeues|RSa|
   SSl|USer|UTil}
```

Description

This command resets general and process-specific counters for encryption and compression to zero. It requires a user with security officer privilege when the switch is in security mode.

The **counter** parameter specifies the category of counters to be reset. The **user**, **util**, **queue** and **jobprocessing** counters display information about the general operation of encryption and compression, while the other counters display information for particular processes.

- If **all** is specified, all counters are reset.
- If **des** is specified, counters for the DES encryption process are reset.
- If **hmac** is specified, counters for the HMAC message process are reset.
- If **jobprocessing** or **queues** is specified, counters for the jobs that have been or are still being processed by the ENCO module are reset.
- If **rsa** is specified, counters for the RSA encryption process are reset.
- If **ssl** is specified, counters for the SSL process are reset.
- If **user** is specified, counters for the interface between ENCO and user applications that use ENCO channels are reset.
- If **util** is specified, counters for the interface between the ENCO module and other user applications that use the ENCO module for one-off jobs are reset.

Examples

To reset all encryption and compression counters, use the command:

```
reset enc cou=all
```

To reset counters for the DES encryption process, use the command:

```
reset enc cou=de
```

Related Commands

show enco counters

set enco key

Syntax

```
SET ENCo KEY=key-id [DESCription=description]
   [IPaddress=ipadd] [MODule=module-id]
```

where:

- *key-id* is a number from 0 to 65535.
- *description* is a character string 1 to 24 characters long. Valid characters are any printable character. If *description* contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.
- *module-id* is the name or number of a switch module (see "Module Identifiers and Names" on page B-2 of Appendix B, Reference Tables for a complete list).

Description

This command changes the user-defined description, IP address, or module for a specific key. It requires a user with security officer privilege when the switch is in security mode.

The **key** parameter specifies the identification number for the key. The specified encryption key must already exist.

The **description** parameter specifies a user-defined description for the key, to make it easier to keep track of different keys.

The **ipaddress** parameter specifies an IP address to associate with the key. The SSH module uses this value to find the RSA public key of a remote peer. Documentation for individual modules specify whether this parameter is required.

The **module** parameter can be used to link a key to a specific module. Documentation for particular modules specify whether this parameter is required.

Examples

To change the description for key 1, use the command:

```
set enc key=1 descr="Switch Z key"
```

Related Commands

create enco key
destroy enco key
show enco key

show enco

Syntax `SHow ENCo`

Description This command displays information about the ENCO module (Figure 24-4, Table 24-1).

Figure 24-4: Example output from the **show enco** command

```
ENCO Module Configuration

  Hardware ............................... NOT PRESENT
  Lowest valid channel ................... 1
  Highest valid channel .................. 511
```

Table 24-1: Parameters in output of the **show enco** command

Parameter	Meaning
Hardware	Encryption hardware, if available.
Lowest valid channel	Identification number of the lowest channel available for use by a user module.
Highest valid channel	Identification number of the highest channel available for use by a user module.

Related Commands

show enco channel
show enco counters

show enco channel

Syntax `SHow ENCo CHannel[=`*`channel`* `[COUnter]]`

where channel is a number from 0 to 512, if the switch has up to 8Mbytes of RAM, 0 to 1024 if the switch has 16Mbytes of RAM, or 0 to 2048 if the switch has 32Mbytes of RAM.

Description This command displays information about active ENCO module channels. If no ENCO channel is specified, a summary of all currently active channels is displayed (Figure 24-5, Table 24-2). If a channel is specified, detailed configuration and status information is displayed about it (Figure 24-6 on page 24-15, Table 24-3 on page 24-15).

If the **counter** parameter is specified, information counters are displayed for the specified channel (Figure 24-7 on page 24-16, Table 24-4 on page 24-16).

Figure 24-5: Example output from the **show enco channel** command

```
Channel  State    User UserID   MDL   pktOverhead Process
-------------------------------------------------------------
   1       UP       SA f0000001 1528    72         DES
   1       UP      SSH 00000001 1584    16         DES
-------------------------------------------------------------
```

Table 24-2: Parameters in output of the **show enco channel** command

Parameter	Meaning
Channel	Channel identification number.
State	Whether the channel is up or down.
User	Whether the user module attached to this channel is TEST or SSH.
UserID	Number used by the user module to identify this channel.
MDL	Maximum data length of packets accepted on this channel.
pktOverhead	Number of bytes that the user module requested be reserved in a packet in front of encoded data.
Process	The process for which the channel is configured: RSA DH DES HMAC

Figure 24-6: Example output from the **show enco channel** command for a specific channel.

```
Channel ....................... 1

Type .......................... ENCODE/DECODE
State ......................... UP
User .......................... SSH
User ID ....................... 00000001
Maximum Data Length ............ 1584
Packet Overhead ............... 16
Process ....................... DES
Process Configuration:
  Des Type..........DES - 56 bit
  Check Type .......NONE
  Channel Type......ENCODE/DECODE
  History Mode......Off
  IV Type...........Random
```

Table 24-3: Parameters in output of the **show enco channel** command for a specific channel

Parameter	Meaning
Channel	Identification number of the channel.
Type	Mode of the channel: ENCODE/DECODE ENCODE ONLY DECODE ONLY
State	Whether the channel is up or down.
User	Whether the user module attached to this channel is TEST or SSH.
User ID	A number used by the user module to identify this channel.
Maximum Data Length	The maximum data length of packets accepted on this channel.
Packet Overhead	The number of bytes reserved at the head of data packets in front of the encoded data, for lower layer packet headers.
Process	The process for which the channel is configured: RSA DES
Process Configuration	Details about a particular process. The fields displayed vary depending on the process.
Max Data Length	The maximum allowed length of data packets on the channel.
Check Type	The type of checksum to be used: XOR8 NONE CRCCCITT
DES Type	[DES] The DES encryption/decryption algorithm used to process packets on the channel: DES–56 bit

Table 24-3: Parameters in output of the **show enco channel** command for a specific channel (Continued)

Parameter	Meaning
Channel Type	[DES] The mode of the channel: ENCODE/DECODE ENCODE ONLY DECODE ONLY.
History Mode	[DES] Whether DES is operating with history mode enabled.
IV Type	[DES] Whether the type of Initialisation Vector (IV) DES uses is zero, random, or specified.
RSA mode	[RSA] Whether the RSA encryption mode on this channel is public or private.

Figure 24-7: Example output from the **show enco channel counter** command

```
Channel Counter:

 UP events                 1     DOWN events               0
 start config              1     attach good               1
 encode NULL packets       0     decode NULL packets       0
 encode bad priorities     0     decode bad priorities     0
 encode bad length         0     decode bad length         0
 encode actions sent       0     decode actions sent       0
 good encodes              0     good decodes              0
 bad encodes               0     bad decodes               0
 reset E actions sent      0     reset D actions sent      0
 good encode resets        0     good decode resets        0
 bad encode resets         0     bad decode resets         0
 discarded encode jobs     0     discarded decode jobs     0
```

Table 24-4: Parameters in output of the **show enco channel counter** command

Parameter	Meaning
UP events	Number of times the channel has entered an up state.
DOWN events	Number of times the channel has entered a down state.
start config	Number of times a configure operation has started on the channel.
attach good	Number of successful attach operations on the channel.
encode NULL packets	Number of encode requests received from a user module with no data packet.
decode NULL packets	Number of decode requests received from a user module with no data packet.
encode bad priorities	Number of encode requests received from a user module with a data packet containing an unknown priority.
decode bad priorities	Number of decode requests received from a user module with a data packet containing an unknown priority.
encode bad length	Number of encode requests received from a user module with a data packet with a bad length.
decode bad length	Number of decode requests received from a user module with a data packet with a bad length.

Table 24-4: Parameters in output of the **show enco channel counter** command (Continued)

Parameter	Meaning
encode actions sent	Number of encode actions which have been sent to the process on this channel.
decode actions sent	Number of decode actions which have been sent to the process on this channel.
good encodes	Number of successful encode operations on the channel.
good decodes	Number of successful decode operations on the channel.
bad encodes	Number of unsuccessful encode operations on the channel.
bad decodes	Number of unsuccessful decode operations on the channel.
reset E actions sent	Number of encode reset actions which have been sent to the process on the channel.
reset D actions sent	Number of decode reset actions which have been sent to the process on the channel.
good encode resets	Number of successful encode resets on the channel.
good decode resets	Number of successful decode resets on the channel.
bad encode resets	Number of unsuccessful encode resets on the channel.
bad decode resets	Number of unsuccessful decode resets on the channel.
discarded encode jobs	Number of encode jobs discarded due to queue overloading or a channel reset.
discarded decode jobs	Number of decode jobs discarded due to queue overloading or a channel reset.

Examples

To show a summary of all active ENCO channels, use the command:

```
sh enc ch
```

To show detailed configuration and status information for channel 1, use the command:

```
sh enc ch=1
```

To show counter information for channel 1, use the command:

```
sh enc ch=1 cou
```

Related Commands

show enco
show enco counters

show enco counters

Syntax

```
SHow ENCo COUnters[={ALl|DEs|HMac|JObprocessing|QUeues|
   RSa|SSl|USer|UTil}]
```

Description

This command displays counters for encryption and compression.

The **counters** parameter specifies the category of counters to display. If a category is not specified, or **all** is specified, all categories of counters are displayed. The **jobprocessing**, **queues**, **user**, and **util** counters display information about the general operation of the ENCO module, while the other parameters display information for particular processes. The default is **all**.

If **des** is specified, counters for the DES encryption process are displayed (Figure 24-8 on page 24-19, Table 24-5 on page 24-19).

If **hmac** is specified, counters for the HMAC message process are displayed (Figure 24-9 on page 24-21, Table 24-6 on page 24-21).

If **jobprocessing** or **queues** is specified, counters are displayed for the jobs that have been or are still being processed by the ENCO module (Figure 24-10 on page 24-21, Table 24-7 on page 24-22).

If **rsa** is specified, counters are displayed for the RSA encryption process (Figure 24-11 on page 24-22, Table 24-8 on page 24-22).

If **ssl** is specified, counters for the SSL process are displayed (Figure 24-12 on page 24-23, Table 24-9 on page 24-23).

If **user** is specified, counters are displayed for the interface between ENCO and user applications that use ENCO channels (Figure 24-13 on page 24-25, Table 24-10 on page 24-26).

If **util** is specified, counters are displayed for the interface between ENCO and user applications that use the ENCO module for one-off jobs (Figure 24-14 on page 24-27, Table 24-11 on page 24-27).

Figure 24-8: Example output from the **show enco counters=des** command

```
ENCO Process DES/3DES Counters
  configGood ..................... 0       configBad ..................... 0
  configNoResource ............... 0       configNotSSH .................. 0
  badBuffer ...................... 0       badAlign ...................... 0
  badLength ...................... 0       nohistory ..................... 0
  desJobs ........................ 0       3Des2KeyJobs .................. 0
  3DesInnerJobs .................. 0       d3DesOuterJobs ................ 0
  noHistJobs ..................... 0       desMacJobs .................... 0
  badDesType ..................... 0       badJobType .................... 0

  unknownJob ..................... 0       error ......................... 0
  reset .......................... 0       confNotDes .................... 0
  commWaitTimeOut ................ 0       dataInWaitTimeOut ............. 0
  dataOutWaitTimeOut ............. 0

  goodDecrypt .................... 0       goodEncrypt ................... 0
  badDecrypt ..................... 0       badEncrypt .................... 0

  DMA1Start ...................... 0       DMA2Start ..................... 0
  DMA1Done ....................... 0       DMA2Done ...................... 0
  DMABed ......................... 0       DMABes ........................ 0
  DMABrkp ........................ 0       DMAConf ....................... 0
  DMA1TimeOut .................... 0       DMA2TimeOut ................... 0
```

Table 24-5: Parameters in output of the **show enco counters=des** command

Parameter	Meaning
configGood	The number of successful channel configurations.
configNoResource	The number of configure attempts without resources.
badBuffer	The number of jobs received by the DES encryption algorithm unit with a bad buffer.
badLength	The number of jobs received by the DES encryption algorithm unit with a bad length (not a multiple of the DES block length).
desJobs	The number of 56-bit DES jobs received by the DES algorithm unit.
3DesInnerJobs	The number of 168-bit 3DES Inner CBC Mode jobs received by the DES/3DES algorithm unit. Not available on AT-8600 switches.
noHistJobs	The number of jobs processed by the DES encryption algorithm unit with history mode set to OFF.
badDesType	The number of jobs received by the encryption algorithm unit with a invalid DES type.
configBad	The number of unsuccessful configuration attempts.
configNotSSH	The number of attempts to configure a software DES channel when the user was not Secure Shell.
badAlign	The number of jobs received by the DES encryption algorithm unit with a bad alignment of the packet.
nohistory	The number of jobs received by the DES encryption algorithm unit without valid history (IV's).
3Des2KeyJobs	The number of 112-bit 3DES jobs received by the DES/3DES algorithm unit. Not available on AT-8600 switches.

Table 24-5: Parameters in output of the **show enco counters=des** command (Continued)

Parameter	Meaning
3DesOuterJobs	The number of 168-bit 3DES Outer CBC Mode jobs received by the DES/3DES algorithm unit. Not available on AT-8600 switches.
desMacJobs	The number of DES-MAC authentication jobs received by the DES/3DES algorithm unit.
badJobType	The number of jobs received by the DES encryption algorithm unit with an invalid job type.
unknownJob	The number of unknown jobs received by the DES encryption algorithm unit.
reset	The number of resets by the hardware encryption unit.
commWaitTimeOut	The number of commands entered for the hardware encryption unit before it was ready for the new command.
dataOutWaitTimeOut	The number of times data was read from the hardware encryption unit before it was ready to output new data.
error	The number of errors that occurred in the DES encryption algorithm unit while processing data.
confNotDes	The number of attempts to configure a DES channel with an invalid encryption type.
dataInWaitTimeOut	The number of times the data was entered to the hardware encryption unit before it is ready for new data.
goodDecrypt	The number of good decryption jobs processed by the DES algorithm unit.
badDecrypt	The number of bad decryption jobs processed by the DES algorithm unit.
goodEncrypt	The number of good encryption jobs processed by the DES algorithm unit.
badEncrypt	The number of bad encryption jobs processed by the DES algorithm unit.
DMA1Start	The number of times the DMA1 channel started.
DMA1Done	The number of times the DMA1 channel completed a transfer.
DMABed	The number of times the Bus Error Destination occurred during DMA transfers.
DMAbrkp	The number of times a DMA break point interrupt occurred.
DMA1TimeOut	The number of times the DMA1 channel timeout occurred.
DMA2Start	The number of times the DMA2 channel started.
DMA2Done	The number of times the DMA2 channel completed a transfer.
DMABes	The number of times a Bus Error Source occurred during DMA transfers.
DMAConf	The number of times a DMA configuration error occurred.
DMA2TimeOut	The number of times the DMA2 channel timeout occurred.
DMA1Start	The number of times the DMA1 channel started.

Figure 24-9: Example output from the **show enco counters=hmac** command

```
ENCO Process HMAC Counters
  goodHashMD5 ..................... 0       badHashMD5 ..................... 0
  goodHashSHA ..................... 0       badHashSHA ..................... 0
  goodConfigure ................... 0       badConfigure ................... 0
  badAlgorithm .................... 0       noResources .................... 0
  badKeyLength .................... 0       unknownJob ..................... 0
  badDataLength ................... 0
```

Table 24-6: Parameters in output of the **show enco counters=hmac** command

Parameter	Meaning
goodHashMD5	The number of good MD5 hashes.
goodHashSHA	The number of good SHA hashes.
goodConfigure	The number of good channel configurations.
badAlgorithm	The number of channel configurations with invalid algorithm types.
badKeyLength	The number of jobs with an invalid key length.
badDataLength	The number of jobs with an invalid data length.
badHashMd5	The number of failed MD5 hashes.
badHashSHA	The number of failed SHA hashes.
badConfigure	The number of failed channel configurations.
noResources	The number of channel configurations with no resources
unknownJob	The number of unknown jobs.

Figure 24-10: Example output from the **show enco counters=jobprocessing** command.

```
ENCO Queues                          Queued  Discarded  Processed
-------------------------------------------------------------------
   Immediate Input queue          0000000000 0000000000 0000000000
   Priority 0 Input queue (high)  0000000000 0000000000 0000000000
   Priority 1 Input queue         0000000000 0000000000 0000000000
   Priority 2 Input queue         0000000000 0000000000 0000000000
   Priority 3 Input queue         0000000000 0000000000 0000000000
   Priority 4 Input queue         0000000000 0000000000 0000000000
   Priority 5 Input queue         0000000000 0000000000 0000000000
   Priority 6 Input queue         0000000000 0000000000 0000000000
   Priority 7 Input queue         0000000000 0000000000 0000000000
   Priority 8 Input queue (low)   0000000000 0000000000 0000000000
   Output queue                   0000000000 0000000000 0000000000
-------------------------------------------------------------------

Input queue length limit ......... 250
Lowest input priority queue ...... 0
Highest input priority queue ..... 0
```

Table 24-7: Parameters in output of the **show enco counters=jobprocessing** command

Parameter	Meaning
ENCO Queues	The internal queues of the ENCO module.
Queued	The current number of jobs in the queue.
Discarded	The number of jobs discarded from the queue.
Processed	The number of jobs processed from the queue.
Immediate Input queue	The queue for jobs required to be done immediately.
Priority *n* Input queue	The prioritized input queue.
Output queue	The output queue.
Input queue length limit	The maximum total length of the input queue.
Lowest input priority queue	The lowest input priority queue with queued actions.
Highest input priority queue	The highest priority queue with queued actions.

Figure 24-11: Example output from the **show enco counters=rsa** command

```
ENCO Process RSA Counters
  goodPublicEncrypt ............... 0       badPublicEncrypt ................ 0
  goodPrivateDecrypt .............. 0       badPrivateDecrypt ............... 0
  goodPrivateEncrypt .............. 0       badPrivateEncrypt ............... 0
  goodPublicDecrypt ............... 0       badPublicDecrypt ................ 0
  goodGenerateKey ................. 0       badGenerateKey .................. 0
  badDataLength ................... 0       badKey .......................... 0
```

Table 24-8: Parameters in output of the **show enco counters=rsa** command

Parameter	Meaning
goodPublicEncrypt	The number of encryption jobs using an RSA public key.
goodPrivateDecrypt	The number of decryption jobs using an RSA private key.
goodPrivateEncrypt	The number of encryption jobs using an RSA private key.
goodPublicDecrypt	The number of decryption jobs using an RSA public key.
goodGenerateKey	The number of RSA keys that have been generated.
badDataLength	The number of jobs with a bad data length.
badPublicEncrypt	The number of failed encryption jobs using an RSA public key.
badPrivateDecrypt	The number of failed decryption jobs using an RSA private key.
badPrivateEncrypt	The number of failed encryption jobs using an RSA private key.
badPublicDecrypt	The number of failed decryption jobs using an RSA public key.
badGenerateKey	The number of failed key generations.
badKey	The number of jobs where the RSA key was invalid.

Figure 24-12: Example output from the **show enco counters=ssl** command

```
ENCO Process SSL Counters
  initialised ..................... 0       initNoResources ................ 0
  configGood ...................... 0       configNoConnection ............. 0
  configBadUserArgs ............... 0       configNoResources .............. 0
  destroyGood ..................... 0       destroyNoConnection ............ 0
  unknownJob ...................... 0       doJobNoConnection .............. 0

Application Data
  appDataEncoded .................. 0       appDataEncodeFail .............. 0
  appDataHmacEncFail .............. 0       appDataDesEncFail .............. 0
  appDataDecoded .................. 0       appDataDecodeFail .............. 0
  appDataDesDecFail ............... 0       appDataHmacDecFail ............. 0

Handshake
  genMasterSecrtGood .............. 0       genKeyMaterialGood ............. 0
  ccsGood ......................... 0       ccsFail ........................ 0
  ccsDesConfigFail ................ 0       ccsHmacConfigFail .............. 0
  processSKEGood .................. 0       processSKENoKey ................ 0
  procsSKECfgRSAFail .............. 0       procsSKERSADecFail ............. 0
  genCKEGood ...................... 0       genCKENoKey .................... 0
  genCKEConfgRSAFail .............. 0       genCKERSAEncFail ............... 0
  processCKEGood .................. 0       processCKENoKey ................ 0
  procsCKECfgRSAFail .............. 0       procsCKERSADecFail ............. 0
  genCVGood ....................... 0       genCVNoKey ..................... 0
  genCVConfigRSAFail .............. 0       genCVRSAEncodeFail ............. 0
  processCVGood ................... 0       processCVNoKey ................. 0
  procssCVCfgRSAFail .............. 0       procssCVRSADecFail ............. 0
  processCVFail ................... 0
```

Table 24-9: Parameters in the output of the **show enco counters=ssl** command

Parameter	Meaning
ENCO Process SSL Counters	
initialized	The number of Initialization operations performed.
configGood	The number of successful channel configurations.
configBadUserArgs	The number of unsuccessful configuration attempts due to bad user arguments.
destroyGood	The number of successful channel configuration de-allocations.
unknownJob	The number of unknown jobs received.
initNoResources	The number of Initialization attempts without resources.
configNoConnection	The number of unsuccessful configuration attempts.
configNoResources	The number of unsuccessful configuration attempts due to lack of resources.
destroyNoConnection	The number of channel configuration de-allocation attempts with no connection.
doJobNoConnection	The number of jobs received with an invalid connection.
Application Data	
appDataEncoded	The number of successful attempts to encode application data.
appDataHmacEncFail	The number of unsuccessful attempts to hash application data.

Table 24-9: Parameters in the output of the **show enco counters=ssl** command

Parameter	Meaning
appDataDecoded	The number of successful attempts to decode application data.
appDataDesDecFail	The number of unsuccessful attempts to decrypt application data.
appDataEncodeFail	The number of unsuccessful attempts to encode application data.
appDataDesEncFail	The number of unsuccessful attempts to encrypt application data.
appDataDecodeFail	The number of unsuccessful attempts to decode application data.
appDataHmacDecFail	The number of unsuccessful attempts to authenticate application data.
Handshake	
genMasterSecrtGood	The number of successful attempts to generate the Master Secret.
ccsGood	The number of successful Change Cipher Spec's processed.
ccsDesConfigFail	The number of unsuccessful Change Cipher Spec's due to failed DES configuration.
processSKEGood	The number of successful Server Key Exchange messages processed.
procsSKECfgRSAFail	The number of failed SKE messages due to failed RSA configuration.
genCKEGood	The number of successful Client Key Exchange messages generated.
genCKEConfgRSAFail	The number of unsuccessful CKE message generations due to failed RSA configuration.
processCKEGood	The number of successful Client Key Exchange messages processed.
procsCKECfgRSAFail	The number of failed CKE messages due to failed RSA configuration.
genCVGood	The number of successful Certificate Verify messages generated.
genCVConfigRSAFail	The number of unsuccessful CV message generations due to failed RSA configuration.
processCVGood	The number of successful Certificate Verify messages processed.
procssCVCfgRSAFail	The number of failed CV messages due to failed RSA configuration.
processCVFail	The number of failed CV messages.
genKeyMaterialGood	The number of successful attempts to generate the key material.
ccsFail	The number of unsuccessful Change Cipher Spec's.
ccsHmacConfigFail	The number of unsuccessful Change Cipher Spec's due to failed HMAC configuration.
processSKENoKey	The number of failed SKE messages due to no key.

Table 24-9: Parameters in the output of the **show enco counters=ssl** command

Parameter	Meaning
procsSKERSADecFail	The number of failed SKE messages due to failed RSA decryption.
genCKENoKey	The number of unsuccessful CKE message generations due to no key.
genCKERSAEncFail	The number of unsuccessful CKE message generations due to failed RSA encryption.
processCKENoKey	The number of failed CKE messages due to no key.
procsCKERSADecFail	The number of failed CKE messages due to failed RSA decryption.
genCVNoKey	The number of unsuccessful CV message generations due to no key.
genCVRSAEncodeFail	The number of unsuccessful CV message generations due to failed RSA encryption.
processCVNoKey	The number of failed CV messages due to no key.
procssCVRSADecFail	The number of failed CV messages due to failed RSA decryption.

Figure 24-13: Example output from the **show enco counters=user** command

```
ENCO User Interface Counters
  startConfig ..................... 0       startReconfig ................... 0
  attachGood ...................... 0       attachFail ...................... 0
  attachNoConfig .................. 0       attachBadUserType ............... 0
  attachedInvalidChan ............. 0       attachedUnusedChan .............. 0
  attachProcNotAvail .............. 0

  reconfigInvalidChan ............. 0       reconfigUnusedChan .............. 0
  reconfigNoConfig ................ 0

  detachInvalidChannel ............ 0       detachUnusedChannel ............. 0
  detachedInvalidChan ............. 0       detachedUnusedChan .............. 0
  detachGood ...................... 0

  decodeInvalidChannel ............ 0       decodeUnusedChannel ............. 0
  encodeInvalidChannel ............ 0       encodeUnusedChannel ............. 0
  codedInvalidChannel ............. 0       codedUnusedChannel .............. 0

  resetInvalidChannel ............. 0       resetUnusedChannel .............. 0
  resetDoneInvalidChan ............ 0       resetDoneUnusedChan ............. 0

  configBadMode ................... 0       configBadUserType ............... 0
  configBadPktLength .............. 0       configBadEncrType ............... 0
  configBadCompType ............... 0       configBadHistoryMode ............ 0
  configBadCheckType .............. 0

  discardInvalidChan .............. 0       discardUnusedChannel ............ 0
```

Table 24-10: Parameters in output of the **show enco counters=user** command

Parameter	Meaning
startConfig	The number of channel configuration requests initiated.
attachGood	The number of successful channel attaches.
attachNoConfig	The number of channel attach requests received with no configuration information.
attachedInvalidChan	The number of channel attached events on invalid channels.
attachProcNotAvail	The number of attaches requesting a process while the process is unavailable.
startReconfig	The number of channel reconfiguration requests started.
attachFail	The number of unsuccessful channel attaches.
attachBadUserType	The number of channel attach requests containing a bad user type.
attachedUnusedChan	The number of channel attached events on unused channels.
reconfigInvalidChan	The number of channel reconfigure requests on invalid channels.
reconfigNoConfig	The number of channel reconfigure requests received with no configuration information.
reconfigUnusedChann	The number of channel reconfigure requests on unused channels.
detachInvalidChannel	The number of channel detach requests on nonexistent channels.
detachedInvalidChan	The number of channel detached events on invalid channels.
detachGood	The number of successful channel detaches.
detachUnusedChannel	The number of channel detach requests on unused channels.
detachedUnusedChan	The number of channel detached events on unused channels.
decodeInvalidChannel	The number of decode requests on nonexistent channels.
encodeInvalidChannel	The number of encode requests on nonexistent channels.
codedInvalidChannel	The number of encode events on nonexistent channels.
decodeUnusedChannel	The number of decode requests on unused channels.
encodeUnusedChannel	The number of encode requests on unused channels.
codedUnusedChannel	The number of encode events on unused channels.
resetInvalidChannel	The number of channel reset requests on nonexistent channels.
resetDoneInvalidChan	The number of channel reset requests on invalid channels.
resetUnusedChannel	The number of channel reset requests on unused channels.
resetDoneUnusedChan	The number of channel reset requests on nonexistent channels.
configBadMode	The number of channel configuration requests containing a bad mode.
configBadPktLength	The number of channel configuration requests containing a bad packet length.

Table 24-10: Parameters in output of the **show enco counters=user** command

Parameter	Meaning
configBadCompType	The number of channel configuration requests containing a bad compression type.
configBadCheckType	The number of channel configuration requests containing a check type.
configBadUserType	The number of channel configuration requests containing a bad user type.
configBadEncrType	The number of channel configuration requests containing a bad encryption type.
configBadHistoryMode	The number of channel configuration requests containing a bad history mode.
discardInvalidChan	The number of discarded jobs on invalid channels.
discardUnusedChannel	The number of discarded jobs on nonexistent channels.

Figure 24-14: Example output from the **show enco counters=util** command

```
ENCO Utility Counters
  codeNullPacket ................. 0       codeBadPacketPriority ........... 0
  codeBadPacketLength ............ 0       codeBadConfig ................... 0
  actionSentEncode ............... 0       actionSentDecode ................ 0
  configureGood .................. 0       configureFail ................... 0
  encodeGood ..................... 0       decodeGood ...................... 0
  encodeBad ...................... 0       decodeBad ....................... 0
```

Table 24-11: Parameters in output of the **show enco counters=util** command

Parameter	Meaning
codeNullPacket	The number of utility jobs where the packet to be processed was null.
codeBadPacketLength	The number of utility jobs where the packet to be processed had a bad packet length.
actionSentEncode	The number of encode jobs started.
configureGood	The number of successful attempts to configure the utility channel.
encodeGood	The number of completed encode jobs.
encodeBad	The number of failed encode jobs.
codeBadPacketPriority	The number of utility jobs where the packet to be processed had a bad priority.
codeBadConfig	The number of utility jobs where the configuration was invalid.
actionSentDecode	The number of decode jobs started.
configureFail	The number of failed attempts to configure the utility channel.
decodeGood	The number of completed decode jobs.
decodeBad	The number of failed decode jobs.

Examples To display counters for RSA encryption, use the command:

```
sh enc cou=rsa
```

To display all encryption and compression counters, use the command:

```
sh enc cou
```

Related Commands
reset enco counter
show enco
show enco channel

show enco debug

Syntax
```
SHow ENCo DEBug
```

Description This command executes a specific sequence of show commands to produce output useful for debugging. The specific commands are:

- **show enco**
- **show enco channel**
- **show enco counters** for all possible parameters

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

Examples To display information useful for debugging, use the command:

```
sh enc deb
```

Related Commands
disable enco debugging
enable enco debugging
reset enco counter
show enco
show enco channel
show enco counters

show enco key

Syntax `SHow ENCo KEY[=key-id]`

where *key-id* is a number from 0 to 65535

Description This command displays information about keys stored in the ENCO key memory. It requires a user with security officer privilege when the switch is in security mode.

If a key identification number is not specified, a summary of all keys stored in key memory is displayed (Figure 24-15, Table 24-12). If an identification number is specified, only the specified key is displayed (Figure 24-16 on page 24-30).

Figure 24-15: Example output from the **show enco key** command

```
ID  Algorithm    Length Digest    Description              Module   IP Address
-------------------------------------------------------------------------------
 0  RSA-PRIVATE   768   E300FFDF  ROUTER KEY               -        -
 1  RSA-PRIVATE   512   DC5014A8  SSH Key                  SSH      -
 2  RSA-PUBLIC   1024   2A1596C9  CHCH router              -        192.168.2.1
 3  RSA-PUBLIC   1024   9348B823  Pauls key               -        -
 4  RSA-PUBLIC    512   C4A396A0  Carls Datafellow key     -        -
 5  DES            64   5A6798BD  Man key for CHCH SA      -        192.168.2.1
 6  GENERAL       768   D56D323F  Auth key for INV         -        -
```

Table 24-12: Parameters in output of the **show enco key** command

Parameter	Meaning
ID	Identification number for the key.
Algorithm	Encryption algorithm for which the key was created: DES RSA-PRIVATE RSA-PUBLIC GENERAL
Length	Length of the key in bytes/bits.
Digest	MD5 digest of the key data.
Description	User-defined description for the key.
Module	Module associated with this key.
IP Address	IP address associated with this key.

Figure 24-16: Example output from the **show enco key** command for a specific DES key

```
ijn69p4v95e5qk
0x425BCFBF55FEC9B8
```

Examples To display the list of all enco keys use the command:

```
sh enc key
```

To display a single DES key use the command:

```
sh enc key=1
```

Related Commands

create enco key
destroy enco key
set enco key