Chapter 50

Link Compression and Encryption

Introduction

This chapter describes the link compression and encryption facilities on the router, and how to configure the Point-to-Point Protocol (PPP), Frame Relay and X.25 link layers to use compression and encryption. PPP, Frame Relay and X.25 support link compression. PPP and Frame Relay also support link encryption using star key management.

The AR450 router supports link compression and link encryption over PPP only.

Overview

Link compression and encryption services have traditionally been provided by external devices connected between a router port and the WAN access device. The disadvantages of external compression and encryption devices are that they require a separate connection to each router port requiring compression or encryption and to each WAN access device. Also, they cannot be managed from within the router's management structure, and they normally do not support dial-up interfaces such as ISDN.

Integrating compression and encryption functions into the router lets a single compression or encryption resource support the compression or encryption of multiple links over any router interface, replacing multiple external devices. Integration also allows the router to support protocols such as PPP multilink, which can spread data from one compression or encryption channel across multiple physical links. The compression and encryption processes can be configured and monitored using the router's own management interface, instead of a separate management system used only for the external device.

Another advantage of integrating both compression and encryption into the router is that a data stream can be compressed and then encrypted without any extra packet processing. Compressing the data stream before it is encrypted can significantly speed up the slower encryption process.

Link compression and encryption operate by compressing or encrypting the whole data stream, including the network layer packet headers used for routing. This means that the packet header is no longer accessible by intermediate routers that do not support the compression or encryption algorithms used. Even if an intermediate router does support the compression or encryption algorithms, packets must be decrypted and decompressed, then compressed and encrypted again at each router so that the packet headers can be read. Consequently, external link compression and encryption is typically used in point-to-point configurations where the local and remote routers are directly connected, without any intermediate routers.

Link Compression

The ENCO module provides multichannel link compression allowing multiple higher layer modules to use compression simultaneously. STAC LZS compression is available in both software and hardware. Predictor compression is available in software. To configure the ENCO module for software or hardware compression, see Chapter 42, Compression and Encryption Services.

Link compression is supported on Frame Relay, PPP and X.25 interfaces. Routing modules such as DECnet, IP and IPX can gain access to the data compression facilities via these link level protocols. PPP and Frame Relay can be configured on a per-interface or per-link basis to use compression. Configuring PPP compression on a per-link basis allows some links in a multilink bundle to be compressed while other links are uncompressed. This has advantages in situations where compression is already being provided on some links, for example on links provided by compressing modems. X.25 is configured on a per-circuit basis.

When a packet is passed to the compression facility for compression, the entire packet following the Frame Relay, PPP or X.25 header is compressed, resulting in a high compression efficiency since the header of any higher layer protocol being carried by these protocols is compressed along with the data. This means that when TCP/IP is being transported over a compressed PPP link, the TCP/IP header is compressed.

PPP

The router implements the Compression Control Protocol (CCP) as defined by RFC 1962 to provide link compression on PPP interfaces. CCP provides a method for negotiating the compression algorithm to use and algorithm-specific parameters such as the check mode. It also provides a mechanism for synchronising the compression histories at each end of the link if they become unsynchronised. The use of STAC LZS and Predictor compression with CCP is defined in RFCs 1962 and 1978, respectively.

PPP commands are used to enable compression for each PPP interface. For example, to create PPP interface 0 over synchronous port 0 and enable STAC LZS compression on the link, use the command:

```
create ppp=0 over=syn0 compression=on compalgorithm=stac
   staccheck=lcb
```

PPP negotiates with the router at the remote end of the link; if both routers have compression enabled on the link and agree on a compression algorithm, compression is used on the link.

During CCP negotiation, the router offers the remote router a choice of compression algorithms that it is prepared to decompress. The router offers a compression algorithm when there are channels configured and available for that algorithm. Compression options that are currently offered are the Predictor compression algorithm using type 1 encapsulation (CCP option 1) and the STAC LZS compression algorithm (CCP option 17). The peer then chooses the option it prefers to compress packets with and informs the offering router of its choice so that it can configure its decompressing channel. When all compression options are rejected by the peer, packets are sent uncompressed.

As part of the negotiation of the STAC LZS option, the router negotiates to add an 8-bit Longitudinal Check Byte (LCB) or an 8-bit sequence number to the data, enabling the packet to be validated during decompression. As part of the negotiation of the Predictor option, the router negotiates to add a Cyclic Redundancy Checksum (CRC) to the data. A check value is useful because PPP does not guarantee reliable, in-order delivery of packets. When a packet is corrupted or lost, its decompression fails when the next packet is received because the compression histories are out of step. Adding a check value allows unsuccessful decompressions to be detected.

To use Software Version 7.4 STAC LZS compression with Software Version 7.2 STAC LZS compression, the check mode on the router running Software Version 7.4 must be set to LCB. To use a COMP-2 coprocessor engine to provide STAC LZS compression, the check mode must be set to LCB at both ends of the PPP link.

When compression histories become unsynchronised while using STAC LZS compression a *Reset Request—Reset Ack* protocol, described in RFC 1962, is used to resynchronise the compression histories. When compression histories become unsynchronised while using Predictor compression a CCP *Configure Request—Configure Ack* exchange is initiated to resynchronise the compression histories.

For more information about configuring PPP link compression see Chapter 15, Point-to-Point Protocol (PPP).

Frame Relay

The router uses the mechanism described in the Frame Relay Forum's *Implementation Agreement 9—Data Compression Over Frame Relay* standard to provide STAC LZS compression on Frame Relay circuits. The router adds an 8-bit Longitudinal Check Byte (LCB) to the data, enabling the packet to be validated on decompression. The negotiation process and compression reset process are based on the PPP CCP mechanism. Compression can be enabled on a per-interface or a per-DLC basis. For example, to create Frame Relay interface 0 over synchronous port 0 and enable compression by default on all DLCs added to the interface, use the command:

```
create fr=0 over=syn0 defcompression=on
```

To create Frame Relay interface 1 over synchronous port 1 with compression disabled by default, and enable compression on DLC 21, use the commands:

```
create fr=1 over=syn1 defcompression=off

add fr=1 dlc=21 compression=on
```

For more information about configuring Frame Relay link compression see Chapter 14, Frame Relay.

X.25

The router uses a simple static configuration process to provide STAC LZS link compression for X.25. X.25 does not reset compression links as it is a reliable transmission protocol. Link compression on X.25 interfaces is configured on a per-circuit basis. For example, to add a MIOX circuit named "RemoteOffice" to X.25T interface 0 and enable compression, use the command:

```
add miox=1 circuit=RemoteOffice pvc=1 compression=on
```

For more information about configuring X.25 compression see Chapter 13, X.25.

Link Encryption

The ENCO module provides multichannel encryption enabling multiple higher layer modules to use encryption simultaneously. DES and two forms of Triple DES encryption are provided in hardware by the ENCO module. To configure the ENCO module for encryption see Chapter 42, Compression and Encryption Services.

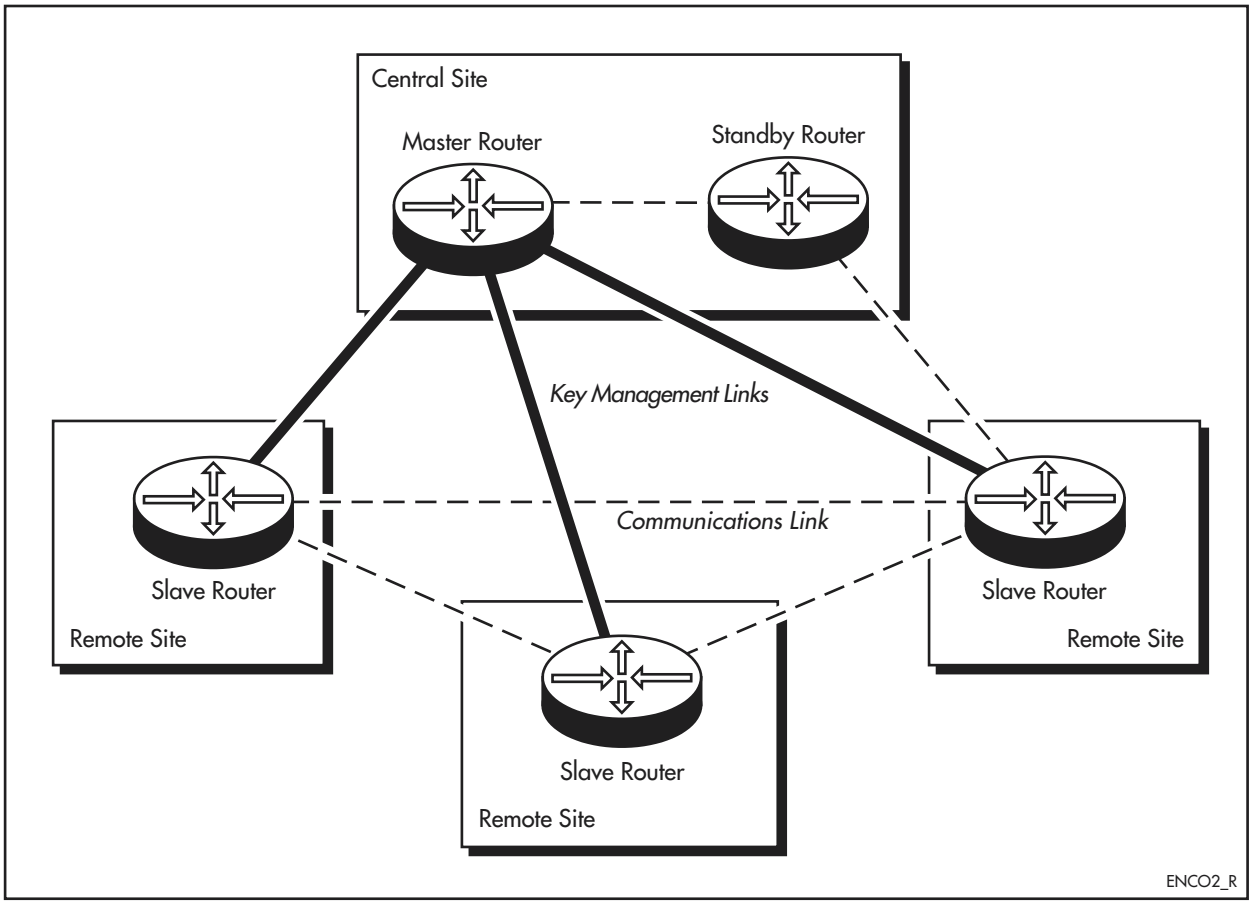
Link encryption is supported on PPP and Frame Relay interfaces. Routing modules such as DECnet, IP and IPX can gain access to the data encryption facilities via these link level protocols. PPP can be configured on a per-interface or per-link basis to use encryption. Configuring encryption on a per-link basis allows some links in a multilink bundle to be encrypted while other links are not encrypted. PPP and Frame Relay use star key management for encrypted links.

When a packet is passed to the encryption facility for encryption, the entire packet following the PPP or Frame Relay header is encrypted. This means that when TCP/IP is being transported over an encrypted PPP link, the TCP/ IP header is encrypted.

Star Key Management

Key management refers to the process of distributing secret key information around a network in a secure manner. PPP and Frame Relay use star encryption key management for link encryption (Figure 50-1).

Figure 50-1: Star encryption key management architecture.



Star key management has the following architecture:

- **One central master router**

  This router is the source of the network key and the master key table. All slaves in the encrypted network must be able to access this router during the master key table transfer.

- **Optional central standby router**

  A router loaded with a network key and master key table in the same way as a slave and can be used to temporarily replace the master router if it fails.

- **A number of remote slave routers**

  Routers configured with the network key and obtain the master key table from the master router.

Keys are stored in flash memory. The encrypted network uses the following:

Component	Description
Session keys	Random number used to DES/3DES encrypt the actual data being transferred across a link. Every time a link is established or a line error occurs, and at random time intervals, a new session key is generated and transferred to the slave router.
Master key table	A table of 160 DES/3DES keys that must be present on every router in the encrypted network. A randomly indexed key from this table is used to encrypt session keys before they are transferred. When a network is installed the master key table is created by the master router from a set of random numbers and distributed to the slave routers. A large table of keys is used, rather than just a single master key, to increase the lifetime of the master key information and therefore reduce the amount of network management required to maintain security.
Network key	Created by the master router and must be entered into every slave router before a master key table download can occur. The network key is used by the master router to encrypt the master key table for transfer to the slave router.

Creating a Star Entity

A star entity is created on any router that is to be part of a star network. When a star user attaches to the ENCO module it must specify an existing star entity to use for its key management requirements. Only one router in the encrypted network can use a master star entity and act as the master router for the network. Other routers act as slaves or standby masters and use star entities configured as slaves or as standbys. To create a star entity, use the command:

```
create star=id mode={master|slave|standby} encalgorithm={des|
   3des2key|3desinner}
```

The standby mode is provided to allow a standby master router to be created. Standby mode operates in the same manner as slave mode, but a standby router can be changed to a master router without erasing the network key and master key table. The standby router loads a master key table from the master router in the same way a slave router does, and is then changed to master mode to create a new master router with the current master key table.

Creating a Network Key

To create a random network key for a master star entity, use the command:

```
set star=id netkey random
```

This command must be entered at least once during the installation of the central master router. It can also be used at a later date to generate a new network key.

Generation of the network key uses random numbers to ensure that a different key is created each time.

Entering the Network Key

After a network key has been created on the master router, it must be entered into the star entities on every slave router in the network. When a new network key is created, it must be entered into slave routers before they can load a new master key table.

To display the network key on the master router, use the command:

```
show star=id netkey
```

The network key is displayed as a line of text characters. To enter it in each slave router, use the command:

```
set star=id netkey value=network key
```

For maximum security, the network key should be entered into slave routers at the central site to avoid transferring it to a remote site in a potentially insecure manner.

The network key is secret information. If it is recorded in any form, extreme care must be taken to ensure that it is irretrievably destroyed or secured from unauthorised access.

Creating the Master Key Table

The master key table is generated from a set of random numbers. A router that is configured in master mode can be used to generate a new set of master keys with the command:

```
set star=id mkt random
```

This command must be entered at least once during the installation of the central master router. It can also be used at a later date to generate a new master key table.

If a new master key table is created on an operational network, data transfer on encrypted links halt after the next line error or timer-created session key exchange occurs. The master table transfer process must be carried out before data transfer can resume.

Master Key Transfer

After the network key has been entered into a slave router, the master key table can be transferred. Each router in the encrypted network needs to have a common set of master keys.

To begin this process the slave and master routers must both have the same network key but have different master key tables. This is the case when the network key has been entered into the slave and either the slave has never had a master key table or a new master key table has been created since the last master key transfer. An operator is required at the central site to carry out this procedure; the remote site does not require operator intervention.

To transfer the master key table

1. Reset the link.

The link to the slave router is reset using the appropriate PPP or Frame Relay command on the master router. This causes the master router to attempt to bring the link back up by sending a session key exchange packet. This packet includes a master key table ID. For example, to reset a PPP link, use the command:

```
reset ppp=n
```

When the slave router receives the session key exchange packet, the slave router compares the master key table ID in the packet to the ID of its own master key table. If the master key table IDs do not match, the slave router sends a request for a new master key table to the master router. The request packet contains the serial number of the slave and a one-way digest of the network key.

When the master router receives the request packet, the serial number is checked against other requests that are currently outstanding. When this is a new request, the digest is checked to ensure the slave is trusted. When the decryption succeeds, the slave has the correct network key and the request is added to the log of pending transfers.

2. Transfer master keys.

To display a list of pending master key table transfer requests, identified by each slave router's serial number, use the following command on the master router:

```
show star mkttransfer log
```

When the appropriate request is seen, enable the transfer of master keys to that slave router by using the command:

```
enable star mkttransfer=serial-number
```

The master router encrypts the master key table with the network key and transmits it to the slave router. When the slave router receives the master key table it uses the network key to decrypt and verify the master key table. The new master key table is then stored in the slave router's key memory.

3. Check the link status.

To monitor the progress of the master key table transfer process on the master router, use the command:

```
show star mkttransfer log
```

Also monitor the state of the PPP or Frame Relay link.

After the master key table transfer is complete and the PPP or Frame Relay link is up, a transfer can be started to the next slave router.

When a master key table has been transferred to a slave router, repeat the process only when the key memory in a slave router is erased or a new master key table is created.

Session Key and IV Exchange

Once the routers at each end of a link have the same master key table, the exchange of session key and initialisation vector is automatic. A *session key* is a random number used as a DES/3DES key in the encryption of data to be sent across a network. The *initialisation vector* (IV) is another random number that sets the initial state of the DES/3DES hardware. Each pair of networked encrypting and decrypting channels has its own session key and IV, which are created by the encrypting channel and sent in encrypted form across the network to the decryption channel. To maximise security the session key and IV are changed often:

- When the link is brought up – this exchange is required to initialise the two ends of the link
- On the occurrence of a line error – rather than reverting to the original session key and IV, a new session key and IV are exchanged
- Once per hour – keys age over time since the more data that is encrypted with them, the easier they are to crack; ensuring that a session key is changed often increases security

The session key exchange process is:

1. At the transmitting/encrypting end of the session, two random numbers are generated; one for the session key, which is used to encrypt and decrypt session data, and the other for the initialisation vector (IV) used to precondition the DES/3DES feedback loops.
2. Another random number is used as an index to select a key from the master key table. This key is used to encrypt the session key and IV. A packet containing the master key index, the encrypted session key and the IV is transmitted to the receiving/decrypting end of the session.
3. On receiving the session key exchange message the decryption channel uses the index to obtain a key from its master key table and then uses this key to decrypt the session key and IV. The encryption and decryption channels are now synchronised and can be used for the transfer of encrypted data.

Standby Master Units

The standby master mode of an encrypting router is a special variation of the slave mode. A standby router operates in the same way as a slave to allow a master key table to be transferred, but the master key table is not erased if the router mode is changed from standby mode to master mode. Allowing a mode change without the loss of the master key table on a standby unit provides a mechanism for transferring the master key table to a new master. This lets the network continue functioning with the currently installed master key table when the master router is removed for servicing.

A master router and any standby routers with current master key tables should be kept secure to prevent an unauthorised transfer of the master key table.

PPP

The router implements the Encryption Control Protocol (ECP) as defined by RFC 1968 to provide encryption on PPP links. ECP provides a method for negotiating the encryption algorithm to use and algorithm-specific parameters. It also provides a mechanism for synchronising the encryption and decryption processes at each end of the link.

PPP commands are used to enable encryption for each PPP interface. For example, to create PPP interface 0 over Ethernet port 0 and enable encryption using star entity 0 on the link, use the command:

```
create ppp=0 over=eth0 encryption=on starentity=0
```

PPP then negotiates with the remote router. If both routers have encryption enabled, then PPP tries to use encryption. If either router is not configured for encryption, or they have different master key tables, then the link is disabled and no data is transferred.

The router currently supports a proprietary version of PPP DES/3DES encryption with star key management that is negotiated using ECP option 0. This proprietary version of DES/3DES uses the negotiation mechanism to provide a secure key management and synchronisation protocol. This protocol is used during network installation to distribute master key tables, and each time a link is brought up or packet corruption occurs, to transfer a new session key.

For more information about configuring PPP link encryption see Chapter 15, Point-to-Point Protocol (PPP).

Frame Relay

The router extends the mechanism described in the Frame Relay Forum's *Implementation Agreement 9—Data Compression Over Frame Relay* standard to provide DES encryption with star key management on Frame Relay circuits. The router adds an 8-bit Longitudinal Check Byte (LCB) to the data, enabling the packet to be validated on decompression. The negotiation process and encryption reset process are based on the PPP ECP mechanism.

For example, to create Frame Relay interface 0 over synchronous port 0 and enable encryption by default on all DLCs added to the interface, use the command:

```
create fr=0 over=syn0 defencryption=on
```

To create Frame Relay interface 1 over synchronous port 1 with encryption disabled by default, and enable encryption on DLC 21, use the commands:

```
create fr=1 over=syn1 defencryption=off

add fr=1 dlc=21 encryption=on
```

For more information about configuring Frame Relay link encryption see Chapter 14, Frame Relay.

Command Reference

This section describes the commands available on the router to configure and monitor link compression and encryption.

The AR450S router supports link compression and link encryption over PPP only.

A user must be logged in with Security Officer privilege to configure encryption services. See Chapter 41, User Authentication for more information about creating users with Security Officer privilege, configuring Remote Security Officers, and logging in with Security Officer privilege. Star encryption key management commands are not allowed in the boot script and are ignored during router startup.

For each interface over which compression or encryption is to be used, a higher layer module must be configured to use compression or encryption. Currently, link compression and encryption is supported on Frame Relay, Point-to-Point Protocol (PPP) and X.25 interfaces. See Chapter 15, Point-to-Point Protocol (PPP) for details of the commands required to enable compression and encryption on a PPP interface. See Chapter 14, Frame Relay for details of the commands required to enable compression and encryption on a Frame Relay interface. See Chapter 13, X.25 for details of the commands required to enable compression on a X.25 interface.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page lxv 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 star

Syntax

```
CREate STAR=entity-id MODe={MASter|SLAve|STAndby}
   ENCalgorithm={DES|3DES2key|3DESInner}
```

where *entity-id* is a number from 0 to 255

Description

This command creates a star entity with a specific identification number. It requires a user with security officer privilege when the router is in security mode.

The **star** parameter specifies the identification number for the star entity. The specified star entity must not already exist.

The **mode** parameter specifies the mode of the star entity. If **master** is specified, the star entity is configured as a master router. If **slave** is specified, the star entity is configured as a slave router. If **standby** is specified, the star entity is configured as a standby router. In a star encryption network, one of the routers in the network is configured as the master router, and the other routers are configured as slave routers. The master router is responsible for generating the system keys and for transferring the keys to slave routers. Standby mode allows a router to behave like a slave router and load the master key table from the master router. After it has the master key table, a standby router can be changed to master mode and used in place of the existing master router.

The **encalgorithm** parameter specifies the encryption algorithm to be used during the transfer of the master key table and session keys. When routers in the network run Software Version 7.4 or earlier, this parameter must be set to DES for backward compatibility.

Examples

To create a star entity with identification number 1, slave mode and standard DES encryption for key transfer, use the command:

```
cre star=1 mod=sla enc=des
```

Related Commands

destroy star
disable star debugging
enable star debugging
enable star mkttransfer
set star
show star
show star counter
show star mkttransfer log
show star netkey

destroy star

Syntax `DESTroy STAR=`*`entity-id`* `[NETkey|MKT]`

where *entity-id* is a number from 0 to 255

Description This command destroys the specified star entity or one of the star entity's constituent components. The memory occupied by the destroyed data is overwritten to ensure the data is irretrievable. This command requires a user with security officer privilege when the router is in security mode.

The **star** parameter specifies the identification number for the star entity. The specified star entity must exist.

The **netkey** parameter specifies that the star entity's network key is to be destroyed.

The **mkt** parameter specifies that the star entity's master key table is to be destroyed.

Destroying a star entity's master key table does not immediately affect the operation of encrypted links using the star entity. However it halts the transfer of data when the next session key exchange is attempted. The transfer of encrypted data does not resume until a new master key table is transferred from the master router to the slave router.

Examples To destroy the network key of a star entity with identification number 1, use the command:

```
dest star=1 net
```

Related Commands

create star
disable star debugging
enable star debugging
enable star mkttransfer
set star
show star
show star mkttransfer log
show star netkey

disable star debugging

Syntax `DISable STAR DEBugging`

Description This command disables the display of debugging information for the star module. Debugging output is sent to the terminal device or Telnet session from which the command was entered. Debugging is disabled by default. This command requires a user with Security Officer privilege when the router is in security mode.

Examples To disable the display of debugging information, use the command:

```
dis star deb
```

Related Commands **enable star debugging**
show star

enable star debugging

Syntax `ENAble STAR DEBugging`

Description This command enables the display of debugging information for the star module. Output is sent to the terminal device or Telnet session from which the command was entered. Debugging is disabled by default. This command requires a user with Security Officer privilege when the router is in security mode.

Examples To enable the display of debugging information, use the command:

```
ena star deb
```

Related Commands **disable star debugging**
show star

enable star mkttransfer

Syntax `ENAble STAR MKTTransfer=`*`sernum`*

where *sernum* is a string 1 to 15 characters long. Valid characters are decimal digits (0–9).

Description This command enables the transfer of a master star entity's master key table to a slave star entity on the router with the specified serial number. This is done during the installation or upgrade of a star encryption network. This command requires a user with security officer privilege when the router is in security mode.

The **mkttransfer** parameter specifies the serial number of the router to which the master key table is to be transferred. Before a transfer can be enabled the master router must have logged a request for a master key transfer from the specified slave router. To determine the serial number of the desired slave router, display the log of master key table transfer requests using the **show star mkttransfer log** command.

Before a transfer is enabled each slave serial number should be checked against a list of known network slave routers to ensure it is a valid member of the encrypted star network.

Examples To enable the transfer of a master key table to a router with the serial number 12341234, use the command:

```
ena star mktt=12341234
```

Related Commands **show star mkttransfer log**

set star

Syntax

```
SET STAR=entity-id {MODe={MASter|SLAve|STAndby}|NETkey
   {RANdom|VALue=value}|MKT RANdom}
```

where:

- *entity-id* is a number from 0 to 255.
- *value* is a string exactly 14 characters long. Valid characters are lowercase letters and digits (2–9). Digits 0 and 1 are illegal, to prevent confusion with the letters O and l.

Description

This command sets either a star entity's mode, network key, or master key table. It requires a user with security officer privilege when the router is in security mode.

The **star** parameter specifies the identification number for the star entity. The specified star entity must exist.

The **mode** parameter specifies the mode of the star entity. If **master** is specified, the star entity is configured as a master router. If SLAVE is specified, the star entity is configured as a slave router. If **standby** is specified, the star entity is configured as a standby router. In a star encryption network, one of the routers in the network is configured as the master router, and the other routers are configured as slave routers. The master router is responsible for the generation of all the system keys and for transferring the keys to the slave routers. Standby mode allows a router to behave like a slave router and load the master key table from the master router. Once it has the master key table a standby router can be changed to master mode, and used in place of the existing master router.

When the mode is changed from **slave** to **master**, the star entity's master key table is destroyed to prevent unauthorised discovery and use of the keys in the master key table.

The **netkey** parameter creates a DES key and assigns the key as the star entity's network key. Either a key consisting of a random value, or a key with the given value is created.

The **random** parameter creates a random network key. The key is used to encrypt the master key table during transfer to a slave star entity. All routers in a star key management network must use the same network key. The key is entered into the star entities on the other routers in the star key management network using the **set star** command.

The **value** parameter creates a network key with the supplied value. This value is an ASCII representation of a known key together with a check value of the key. The value must be the output of the **show star netkey** command or the check value will be incorrect. This parameter is used to enter a network key, previously created with a **create star** command, into the star entities on the other routers in the star key management network.

Changing a star entity's network key by using the **set star** command does not affect the operation of encrypted links using the star entity. However, it prevents the transfer of future master key tables for the star entity to or from the router when the network key is different to that of the other star entities.

The **mkt** parameter creates a table of 160 DES keys and assigns the table as the star entity's master key table. The table consists of random values.

Only the master key table of a master star entity can be set with this command. A slave star entity is sent the master key table created by its master entity.

Changing a star entity's master key table does not immediately affect the operation of encrypted links using the star entity, but it halts the transfer of data when the next session key exchange is attempted. The transfer of encrypted data does not resume until a new master key table is transferred from the master router to the slave router.

Examples

To change the mode of a star entity with the identification number 1 from "standby" to "master", use the command:

```
set star=1 mod=mas
```

To set the network key of a newly created slave star entity 1 to that of its master, use the command:

```
set star=1 net val=lb865qh6pscmem
```

To create a new master key table for a master star entity 1, use the command:

```
set star=1 mkt ran
```

Related Commands

create star
destroy star
disable star debugging
enable star debugging
enable star mkttransfer
set star
show star
show star counter
show star mkttransfer log
show star netkey

show star

Syntax SHow STAR[=*entity-id*]

where *entity-id* is a number from 0 to 255

Description This command displays information about star entities. It requires a user with security officer privilege when the router is in security mode.

If a star entity is not specified, summary information for all star entities is displayed (Figure 50-2). If a star entity is specified, detailed information about the specified star entity is displayed (Figure 50-3, Table 50-1).

Figure 50-2: Example output from the **show star** command.

```
Star Entities

      0
      1
```

Figure 50-3: Example output from the **show star** command for a specific star entity

```
 Star ID ........................... 0
 Star Mode ......................... MASTER
Key Transfer Encryption Algorithm. 3DES - 112 bit - outer CBC
 Netkey Valid ...................... FALSE
 Master Key Table Valid ............ FALSE
 Netkey Digest ..................... 00000000
 Master Key Table Digest ........... 00000000
 Transfer State .................... INITIAL
```

Table 50-1: Parameters in output of the **show star** command for a specific star entity

Parameter	Meaning
Star ID	Identification number of the star entity.
Star Mode	Whether the mode of the star entity is master, slave, or standby.
Key Transfer Encryption Algorithm	Encryption/decryption algorithm used for key transfers for this star entity: DES - 56 bit 3DES - 112 bit - outer CBC 3DES - 168 bit - inner CBC
Netkey Valid	Whether the star entity's network key is valid.
Master Key Table Valid	Whether the star entity's master key table is valid.
Netkey Digest	MD5 digest of the star entity's network key.
Master Key Table Digest	MD5 digest of the star entity's master key table.
Transfer State	Whether the state of the master key table transfer is initial, enabled, disabled, or unknown.

Examples To display a list of all star entities stored in key memory, use the command:

```
sh star
```

To display detailed information about star entity 0, use the command:

```
st star=0
```

Related Commands

create star
destroy star
disable star debugging
enable star debugging
enable star mkttransfer
set star
show star counter
show star mkttransfer log
show star netkey

show star counter

Syntax `SHow STAR[=`*`entity-id`*`] COUnter`

where *entity-id* is a number from 0 to 255

Description This command displays counters for the star module, and requires a user with Security Officer privilege when the router is in security mode.

If a star entity is not specified, general counters are displayed (Figure 50-4, Table 50-2 on page 50-22). If a star entity is specified, counters for that entity are displayed (Figure 50-5 on page 50-24, Table 50-3 on page 50-24).

Figure 50-4: Example output from the **show star counter** command.

```
General Star Counters

  attachNoConfig             0   attachBadId              0
  detachInvalidChannel       0   detachUnusedChannel      0
  setCompInvalidChannel      0   setCompUnusedChannel     0
  resetInvalidChannel        0   resetUnusedChannel       0
  resetENoStar               0   resetDNoStar             0
  encodeInvalidChannel       0   decodeInvalidChannel     0
  encodeUnusedChannel        0   decodeUnusedChannel      0
  encoEventUnusedChannel     0   encoEventStarUnused      0

  getMKTUnusedChannel        0   getMKTInvalidChannel     0
  getMKTGood                 0   getMKTNoStar             0
  setMKTUnusedChannel        0   setMKTInvalidChannel     0
  setMKTGood                 0   setMKTNoStar             0

  getSKUnusedChannel         0   getSKNoStar              0
  setSKUnusedChannel         0   setSKNoStar              0
  getInfoUnusedChannel       0   getInfoNoStar            0

  transferTimeoutChanUnused  0   transferTimeoutNoStar    0
  transferRequestChanUnused  0   transferRequestNoStar    0
  transferEnableChanUnused   0   transferEnableNoStar     0
  transEndNotifyChanUnused   0   transferEndNotifyNoStar  0
  sessTimeoutUnusedChannel   0   sessTimeoutNoStar        0
  createStar                 1   destroyStar              0
  destroyNetKey              0   destroyMKT               0
  setStarMode                0   setStarEncAlgorithm      0
  setStarNetKey              0   setStarMKTRandom         0
```

Table 50-2: Parameters in output of the **show star counter** command

Parameter	Meaning
General Star Counters	Counters for star encryption key management.
attachNoConfig	The number of channel attach requests received with no configuration information.
attachBadId	The number of unsuccessful attempts to attach to an ENCO channel by a star user due to the specification of an invalid star entity.
detachInvalidChannel	The number of attempts to detach from a nonexistent channel.
detachUnusedChannel	The number of attempts to detach from an unused channel.
setCompInvalidChannel	The number of attempts to enable/disable compression on a nonexistent star channel.
setCompUnusedChannel	The number of attempts to enable/disable compression on an unused star channel.
resetInvalidChannel	The number of attempts to reset a nonexistent star channel.
resetUnusedChannel	The number of attempts to reset an unused star channel.
resetENoStar	The number of attempts to reset an non-star encoding channel by a star user.
resetDNoStar	The number of attempts to reset an non-star decoding channel by a star user.
encodeInvalidChannel	The number of encode jobs passed to the star module with invalid channel identifiers.
decodeInvalidChannel	The number of decode jobs passed to the star module with invalid channel identifiers.
encodeUnusedChannel	The number of encode jobs passed to the star module with the identifier of an unused channel.
decodeUnusedChannel	The number of decode jobs passed to the star module with the identifier of an unused channel.
encoEventUnusedChannel	The number of events from the ENCO module for an unused channel.
encoEventStarUnused	The number of encode from the ENCO module for an unused star entity.
getMKTUnusedChannel	The number of get master key table requests on an unused channel.
getMKTInvalidChannel	The number of get master key table requests for a nonexistent channel.
getMKTGood	The number of successful attempts to get a master key table.
getMKTNoStar	The number of get master key table requests with no star.
setMKTUnusedChannel	The number of set master key table requests on an unused channel.
setMKTInvalidChannel	The number of set master key table events for a nonexistant channel.
setMKTGood	The number of successful attempts to set a master key table.
setMKTNoStar	The number of set master key table requests with no star.

Table 50-2: Parameters in output of the **show star counter** command (Continued)

Parameter	Meaning
getSKUnusedChannel	The number of get session key requests on an unused channel.
getSKNoStar	The number of get session key requests with no star.
setSKUnusedChannel	The number of set session key requests on an unused channel.
setSKNoStar	The number of set session key requests with no star.
getInfoUnusedChannel	The number of get Info requests on an unused channel.
getInfoNoStar	The number of get Info requests with no star.
transferTimeoutChanUnused	The number of star master key table transfer timeouts on unused channels.
transferTimeoutNoStar	The number of star master key table transfer timeouts on non-star channels.
transferRequestChanUnused	The number of star master key table transfer requests on unused channels.
transferRequestNoStar	The number of star master key table transfer requests on non-star channels.
transferEnableChanUnused	The number of star master key table transfer enables on unused channels.
transferEnableNoStar	The number of star master key table transfer enables on non-star channels.
transEndNotifyChanUnused	The number of star master key table transfer end notifies on unused channels.
transferEndNotifyNoStar	The number of star master key table transfer end notifies on non-star channels.
sessTimeoutUnusedChannel	The number of session key timeouts on unused channels.
sessTimeoutNoStar	The number of session key timeouts on non-star channels.
createStar	The number of successful attempts to create a star.
destroyStar	The number of successful attempts to destroy a star.
destroyNetkey	The number of successful attempts to destroy the network key component of a star entity.
destroyMKT	The number of successful attempts to destroy the master key table component of a star entity.
setStarMode	The number of successful attempts to set the mode of the star.
setStarEncAlgorithm	The number of successful attempts to set the encryption algorithm of the star.
setStarNetKey	The number of successful attempts to set the NetKey of the star.
setStarMKTRandom	The number of successful attempts to create a new random MKT.

Figure 50-5: Example output from the **show star counter** command for a specific star entity.

```
encrConfGood              0     compConfGood            0
 confGood                 0     compConfGoodDetachE     0
 encrConfFail             0     compConfFail            0
 confFail                 0     compConfFailDetachE     0
 encrDetached             0     compDetached            0
 detached                 0     detachedConfFail        0
 encodeGood               0     encodeFailed            0
 decodeGood               0     decodeFailed            0
 encodeDiscarded          0     decodeDiscarded         0
 resetGetSKAlreadyCurr    0     resetSetSKAlreadyCurr   0
 resetDNoSessionKey       0     resetDBadMKTIndex       0
 resetCompHist            0

 getMKTAlreadyCurrent     0     getMKTStarInvalid       0
 getMKTEncodeFailed       0     getMKTGood              0
 setMKTNULLBuffer         0     setMKTAlreadyCurrent    0
 setMKTNoStar             0     setMKTDecodeFailed      0
 setMKTGood               0

 getSKAlreadyCurrent      0     getSKMKTInvalid         0
 getSKEncodeFailed        0     getSKResetFailed        0
 getSKGood                0
 setSKNULLBuffer          0     setSKAlreadyCurrent     0
 setSKDecodeFailed        0     setSKMKTInvalid         0
 setSKResetFailed         0     setSKGood               0
```

Table 50-3: Parameters in output of the **show star counter** command for a specific star entity

Parameter	Meaning
resetGetSKAlreadyCurr	The number of reset star entity get session key operations requested while another get session key operation is already being processed.
resetSetSKAlreadyCurr	The number of reset star entity set session key operations requested while another set session key operation is already being processed.
resetDNoSessionKey	The number of reset decoding star entity requests received from the user module with no session key packet.
resetDBadMKTIndex	The number of reset decoding star entity requests received from the user module with a bad master key table index.
resetCompHist	The number of times the compression history has been reset.
getMKTAlreadyCurrent	The number of get master key table operations requested while another get master key table operation is already being processed.
getMKTStarInvalid	The number of get master key table requests received from the user module with an invalid star entity specified.
getMKTEncodeFailed	The number of get master key table operations in which the encoding of the master key table was unsuccessful.
getMKTGood	The number of successful get master key table operations.
setMKTNULLBuffer	The number of set master key table requests received with no master key table buffer.

Table 50-3: Parameters in output of the **show star counter** command for a specific star entity (Continued)

Parameter	Meaning
setMKTAlreadyCurrent	The number of set master key table operations requested while another set master key table operation is already being processed.
setMKTNoStar	The number of set master key table requests received from the user module with no star entity.
setMKTDecodeFailed	The number of set master key table operations in which the decoding of the master key table was unsuccessful.
setMKTGood	The number of successful set master key table operations.
getSKAlreadyCurrent	The number of get session key operations requested while another get session key operation is already being processed.
getSKMKTInvalid	The number of get session key requests specifying a star entity with an invalid master key table.
getSKEncodeFailed	The number of get session key operations in which the encoding of the session key was unsuccessful.
getSKResetFailed	The number of get session key operations in which the reset of the channel was unsuccessful.
getSKGood	The number of successful get session key operations.
setSKNULLBuffer	The number of set session key requests received with no session key buffer.
setSKAlreadyCurrent	The number of set session key operations requested while another set session key operation is already being processed.
setSKDecodeFailed	The number of set session key operations in which the decoding of the session key was unsuccessful.
setSKMKTInvalid	The number of set session key requests specifying a star entity with an invalid master key table.
setSKResetFailed	The number of set session key operations in which the reset of the channel was unsuccessful.
setSKGood	The number of successful set session key operations.

Examples

To display general counters, use the command:

```
sh star cou
```

To display detailed counters for star entity 0, use the command:

```
sh star=0 cou
```

Related Commands

create star
destroy star
disable star debugging
enable star debugging
enable star mkttransfer
set star
show star
show star mkttransfer log
show star netkey

show star mkttransfer log

Syntax `SHow STAR MKTTransfer LOG`

Description This command displays the log of star master key table transfer requests. It requires a user with security officer privilege when the router is in security mode. (Figure 50-6, Table 50-4).

Figure 50-6: Example output from the **show star mkttransfer log** command

```
Star Master Key Table transfer request log:
Serial Number  StarID User UserID   State     Time     Date        Requests
-------------------------------------------------------------------------
12138430            1  PPP        0 RECEIVED  11:19:28 14-Apr-1997          6
-------------------------------------------------------------------------
```

Table 50-4: Parameters in output of the **show star mkttransfer log** command

Parameter	Meaning
Serial Number	Serial number of the router that is requesting a master key table.
StarID	Identification number of the star entity whose master key table is being requested.
User	User of the channel on which the request was received.
UserID	User ID of the channel on which the request was received.
State	State of the request: Received Badkey Sending Failed Completed
Time	Time of the first request.
Date	Date of the first request.
Requests	Number of identical requests received.

Examples To display the log of master key transfer requests, use the command:

```
sh star mktt log
```

Related Commands

create star
destroy star
disable star debugging
enable star debugging
enable star mkttransfer
set star
show star
show star counter
show star netkey

show star netkey

Syntax `SHow STAR=`*`entity-id`* `NETkey`

where *entity-id* is a number from 0 to 255

Description This command displays an ASCII representation of the network key for a specific star entity (Figure 50-7). This command displays the network key of a master star entity, and requires a user with Security Officer privilege when the router is in security mode.

Figure 50-7: Example output from the **show star netkey** command

```
kducq9iy9ueosk
```

Examples To display the network key for star entity 0, use the command:

```
sh star=0 net
```

Related Commands

create star
destroy star
disable star debugging
enable star debugging
enable star mkttransfer
set star
show star
show star counter
show star mkttransfer log