Chapter 45

IP Security (IPsec)

Introduction

This chapter describes:

- Internet Protocol Security Facility (IPsec)
  A set of security protocols that provides encryption and authentication to IPv4 and IPv6 packets
- Internet Security Association Key Management Protocol (ISAKMP)
  A common framework for a key management implementation – the framework that IKE uses
- Internet Key Exchange (IKE) protocol
  A mechanism for negotiating IPsec protection and keys.
- NAT-Traversal
  An enhancement to IPsec and ISAKMP protocols that lets Virtual Private Network (VPN) clients communicate through NAT gateways over the Internet.
- Configuration examples of VPN client support for Microsoft Windows 2000 and XP®

Additionally, this chapter describes an implementation of Security Associations in Software Release 7.4 (based on 1998 Internet Drafts), and how to upgrade a Virtual Private Network (VPN) based on this IPsec implementation.

Some interface and port types mentioned in this chapter may not be supported on your router. The interface and port types that are available vary depending on your product's model, and whether an expansion unit (PIC, NSM) is installed. For more information, see the Hardware Reference.

IP Security (IPsec)

IPsec provides the following security services for traffic at the IP layer:

- Data origin authentication – identifying who sent the data.
- Confidentiality (encryption) – ensuring that the data has not been read en route.
- Connectionless integrity – ensuring the data has not been changed en route.
- Replay protection – detecting packets received more than once to help protect against denial of service attacks.

IPsec protects one or more paths between a pair of hosts, a pair of security gateways, or a security gateway and a host. A *security gateway* is an intermediate device, such as a router or firewall, that implements IPsec. Two devices that use IPsec to protect a path between them are called *peers*.

Security Protocols and Modes

The following protocols provide security services:

- Encapsulating Security Payload (ESP)
- Authentication Header (AH)

The Encapsulating Security Payload protocol (protocol 50) provides encryption (confidentiality) and authentication (connectionless integrity and data origin authentication). ESP protects an IP packet but not additional headers that ESP adds.

The Authentication Header protocol provides everything that ESP does – connectionless integrity and data origin authentication – but it does not encrypt to ensure confidentiality. AH protects an IP packet and also additional headers that AH adds.

ESP and AH may be applied alone or together to provide the desired security services. The security they provide depends on the cryptographic algorithms they apply. Both are algorithm-independent, which lets new algorithms be added without affecting other parts of the implementation. A set of default algorithms is specified by the RFCs to ensure interoperability between IPsec implementations.

Both protocols support the two IPsec modes: transport and tunnel. Transport mode protects an IP load in upper layer protocols such as UDP and TCP protocols. AH and ESP intercept packets from the transport layer that are intended for the network layer, protect the transport header, and provide the configured security. Transport mode provides end-to-end security where the communications endpoint is also the cryptographic endpoint.

Tunnel mode protects data by encapsulating entire packets that are decapsulated by a security gateway. Tunnel mode may be used with security gateways where the destination of the packet differs from the security endpoint. IPsec also supports nested tunnels.

Compression Protocol

IP Payload Compression (IPComp) provides compression services for traffic at the IP layer. IPComp is specified by the following RFCs:

- RFC 2393, "IP Payload Compression Protocol (IPComp)"
- RFC 2395, "IP Payload Compression Using LZS"

IPComp provides *stateless compression* to compressible IP packets. Stateless compression means that each packet is compressed (and decompressed) with no dependence on any other packet. Stateless compression is not as efficient as *stateful compression* that uses information gained in compressing previous packets to better compress the current packet. However, stateless compression is required for IP packets as they may be re-ordered or lost while traversing the Internet, thus depriving the decompressing device of the information used by the compressing device to achieve the more efficient compression.

The compression algorithms that IPComp uses can expand uncompressible data so IPComp tries to compress packets when they are larger than a certain size. This increases the chance of achieving a useable compression. IPComp sends packets uncompressed if they expand during compression. When too many uncompressible packets are seen, IPComp stops trying to compress them for a set number of packets.

IPComp must try to compress packets before IPsec because encrypted data is not compressible.

Security Associations (SA)

A Security Association (SA) is a one-way connection that provides security services between IPsec peers. For example, SAs determine the security protocols and the keys. An IPsec database (SADB) maintains the SAs that the IPsec protocols use.

Typically, two SAs are created on each end for a traffic stream. That is, Host A would have one SA to process inbound traffic and one to process outbound traffic. Individual SAs are protocol-specific; each uses Authentication Header (AH) or Encapsulating Security Payload (ESP) protocols but not both. When multiple SAs are necessary for a traffic stream between two hosts, the collection of SAs is grouped into an *SA bundle*.

An SA is uniquely identified by a combination of the following:

- a random number called the Security Parameter Index (SPI)
- an IP destination address
- a security protocol header, either AH or ESP.

A transport mode SA is a security association between two hosts. Tunnel mode protects the whole IP packet by applying AH or ESP to tunnelled packets.

A tunnel mode SA is one applied to an IP tunnel. Whenever either end of a security association is a security gateway, the SA **must** be in tunnel mode. This avoids problems with fragmentation and reassembly of IPsec packets, and avoids problems where multiple paths exist between security gateways. An exception is where the security gateway is acting as a host and the traffic (e.g. a Telnet session) is destined for a security gateway.

A tunnel mode SA has an outer IP header that specifies the IPsec peer, and an inner IP header that specifies the destination for the packet. The security protocol header appears after the outer IP header and before the inner IP header. If AH is employed in tunnel mode, the tunnelled packet and portions of the outer IP header are protected. If ESP is employed, only the tunnelled packet is protected.

ISAKMP/IKE

The Internet Security Association and Key Management Protocol (ISAKMP) provides a framework for negotiating SAs and their attributes, including secret keys. The Internet Key Exchange (IKE) protocol is based on the ISAKMP framework and provides an authenticated key exchange method using the Diffie-Hellman algorithm.

ISAKMP

The Internet Security Association and Key Management Protocol (ISAKMP) is defined in RFC 2408. It prescribes procedures and packet formats to securely establish, negotiate, modify, and delete Security Associations (SAs) and their attributes, including secret keys. It provides a common framework for a key management implementation and is independent of key generation, authentication, encryption algorithms, and SA definitions. A key management protocol based on the ISAKMP framework must define its own key generation techniques, encryption algorithms, and authentication types. ISAKMP can be implemented over any transport layer although UDP port 500 is required in order to comply with the RFC.

ISAKMP uses random values called *cookies* to identify separate negotiations and to stop denial of service attacks. Although total protection against denial of service attacks is impossible, it is possible to protect computing resources such as slow Diffie-Hellman calculations. ISAKMP requires that the authentication, key exchange, and SA negotiation be linked together. *Connection hijacking*, where a third party switches places with one of the original parties during the negotiation, is therefore not possible. Man-in-the-middle attacks such as interception, insertion, deletion, modification, and replay are prevented by message authentication. If anything out of the ordinary happens during an ISAKMP exchange, the ISAKMP state machine returns to idle state and does not create unauthenticated SAs.

ISAKMP Messages

Each ISAKMP message is constructed from a generic ISAKMP header followed by one or more payloads. The ISAKMP RFC defines the following payloads:

- Security Association (SA) – used to negotiate security attributes and indicates the DOI and situation under which the negotiation is taking place.
- Proposal (P) – part of the SA payload containing security protocols such as AH and ESP
- Transform (T) – part of the proposal payload containing security attributes for the specified security mechanism
- Key Exchange (KE) – key exchange specific data

- Identification (ID) – DOI-specific information for identifying communicating peers
- Certificate (CERT) – used to transport certificates
- Certificate Request (CR) – used to request a certificate from the other peer
- Hash (HASH) – data generated by the hash function for the purpose of verifying the integrity of the message or entity authentication
- Signature (SIG) – data generated by the digital signature function used to authenticate some part of the message
- Nonce (NONCE) – random data used to guarantee liveness and protect against replay attacks
- Notification (N) – ISAKMP and DOI-specific informational data
- Delete (D) – used to notify the other peer that a SA has been deleted
- Vendor ID (VID) – used to enable devices to recognise remote instances of their implementations and use proprietary extensions
- Attribute (ATTR) – used to exchange configuration information

ISAKMP requires that an authenticated key exchange be supported and that strong peer authentication be used. However it does not define what algorithms must be used.

Key Management

Encryption algorithms are used with encryption keys. Encryption algorithms such as DES have undergone hundreds of hours of cryptanalysis and the mathematics behind the algorithm are well published. It is unlikely that an attack on DES encrypted data would come in the form of an attack on the DES encryption algorithm. It is more likely that an attacker would try to discover the value of the encryption key used to encrypt the data. This means that the security of encrypted data depends on the security of the encryption key. If a key has been created and securely stored, an intruder would have to try every possible key, which is unlikely.

The term *key management* refers to the creation, distribution, storage, and deletion of keys. If an attacker were to obtain an encryption key, they could decrypt sessions they recorded any length of time ago. Therefore a key management protocol must ensure keys are never compromised.

The concept of a *session key* is used to increase the security of encryption keys. If a key is changed frequently, an attacker could not try every possible key often enough on the encrypted data. Also, if a key is compromised in the future, only the data encrypted with that particular session key can be decrypted.

Session keys must be changed on a regular basis, so manually keyed IPsec soon becomes unmanageable. Keys to other sites must be changed regularly for every site in the VPN. This can be a long and tedious job in large VPNs. Therefore, session keys must be changed automatically, and more importantly, securely.

Two devices that engage in ISAKMP negotiations are described as being ISAKMP *peers*. Communication between two ISAKMP peers takes the form of an ISAKMP exchange, which consists of a finite number of messages and is dynamically created and destroyed as required. ISAKMP defines three negotiation phases for key management:

Phase 1 is when the ISAKMP SA is established that provides a secure, authenticated channel for traffic between two ISAKMP peers.

Phase 1.5 exchanges occur after the ISAKMP SA has been created in phase 1, but before phase 2 SAs can be created. These are commonly used for extended authentication and remote configuration.

Phase 2 is when all traffic is under the protection of the ISAKMP SA. The negotiation of SAs and keys on behalf of services such as IPsec, as defined in a particular DOI, makes it possible to negotiate more than one phase 2 SA over the same ISAKMP SA without having to re-establish communications with the ISAKMP peers.

Key Exchange

The most important property of a key management protocol is how session keys are exchanged. Key exchange has two parts:

- Key generation
- Key transport

A key exchange protocol must ensure that only the two negotiating parties know the value of the exchanged session key. SKIP, Photuris, SKEME, and Oakley are examples of key exchange protocols. If one key is compromised, it must not cause other keys to be compromised. This property is called Perfect Forward Secrecy (PFS). Identity protection ensures that the identity of the two parties exchanging keys cannot be discovered. If either party can initiate the key exchange, then the key exchange protocol is said to display *symmetry*. Other properties such as replay protection ensure that the key management protocol is not susceptible to denial of service and man-in-the-middle attacks.

The principal behind public key cryptography is that a user can encrypt a message with a public key and be sure that only the person who holds the private part of the public key pair can decrypt it. This makes public key cryptography ideal for transporting session keys between IPsec peers.

RSA is a public key algorithm that is commonly used in key exchanges. A session key can be created at one end of a connection, encrypted with the peer's RSA public key, and transmitted over the connection to the peer. Because only the peer who has the private part of the RSA key can decrypt the session key, the key can be securely transferred. The Secure Shell protocol is an example of using RSA to exchange session keys.

The Diffie-Hellman key exchange method also uses public key cryptography. It is not an encryption algorithm and is designed specifically for key exchange. The public and private values in Diffie-Hellman do not need to be kept secret and commonly used values are well documented and published. The mathematics behind Diffie-Hellman enables two parties to generate a shared secret key with the knowledge that no other party can calculate that same value.

Using RSA for key exchange is faster than Diffie-Hellman but has the disadvantage that the public part of the RSA keys must first be exchanged. That the other party in the negotiation is the true owner of the key must also be verified. Diffie-Hellman does not require that information be previously exchanged but is slightly slower than RSA.

See Chapter 25, Compression and Encryption Services for more information on public key cryptography, RSA, and Diffie-Hellman.

Key Exchange Authentication

Key exchange authentication guarantees that each party knows they are talking to the correct party. It must also ensure that all messages before and after the authentication procedure could not have been substituted by a third party. This means that all messages must be authenticated.

Key exchange without authentication is susceptible to an attack known as the "man-in-the-middle attack" where a third party intercepts the key negotiation and pretends to be the opposite end to both parties. The result is that both parties negotiate keys with the third party thinking they have negotiated with each other.

Authentication of each entity can take place after the session key has been generated or during the key exchange process itself. It is common for the session key to include a piece of information that could be known only by the actual negotiating parties. Then each party can be authenticated by proving they know the session key.

Authentication techniques usually fall into two categories—weak and strong. Weak authentication includes sending keys in clear-text or using easily guessed keys. Strong authentication includes public key encryption and digital signatures. Strong authentication techniques generally require more overhead.

Password (or shared key) authentication works on the principal that the remote party is the only one who knows the value of the shared secret. Both parties are authenticated when they prove they know the secret. For this kind of authentication to work, the shared secret must be securely entered at both ends of the link.

Public key cryptography is an example of strong authentication. If a remote party decrypts a message encrypted with a particular public key, then they must be the true owner of the public/private key pair. Before authentication can take place, public keys must first be exchanged and verified as the correct key for the other party. This can be done manually by the administrator (by comparing the key at each peer), or automatically by using certificates that link a public key to an entity and are signed by a trusted third party. Because public key values are not secret, the key exchange (or certificate exchange) does not need to be done over a secure channel.

ISAKMP Exchange Types

The ISAKMP exchange and ISAKMP messages are the building blocks for a key management protocol using the ISAKMP framework. An ISAKMP exchange consists of a finite number of messages that are constructed from a defined set of payloads. The following exchange types are defined in the ISAKMP RFC:

- Base
- Identity protection
- Authentication
- Aggressive
- Informational

A key management protocol that uses ISAKMP such as IKE defines additional exchange types as necessary.

IKE

The Internet Key Exchange (IKE) protocol is defined in RFC 2409 and provides key management for IPsec by combining parts of the Oakley and SKEME protocols with the ISAKMP framework.

The Oakley key determination protocol is defined in RFC 2412 and describes a method of authenticated key exchange with PFS and identity protection. The SKEME protocol is defined in Krawcyzk's document "*SKEME: A Versatile Secure Key Exchange Mechanism for Internet*". It also provides an authenticated key exchange and describes a method of public key authentication with fast re-keying using nonces.

IKE Exchanges

IKE defines two phase 1 exchanges to create the ISAKMP SA:

- *Main mode* is an implementation of the ISAKMP identity protection exchange and must be implemented.
- *Aggressive mode* is an implementation of the ISAKMP aggressive exchange and may optionally be implemented.

IKE also defines a *New Group mode* that must be used after phase 1, and allows new Diffie-Hellman public key values to be negotiated.

IKE defines a new ISAKMP exchange called *Quick mode* for phase 2 only. This exchange negotiates SAs on behalf of other services such as IPsec.

ISAKMP informational exchanges can take place in phase 1 or phase 2 and notify the other end of errors and SA deletions.

To allow the exchange of configuration information, a *transaction exchange* has been defined. This exchange can take place in either phase 1, phase 1.5, or phase 2. It is commonly used in phase 1.5 to transport extended authentication information.

All IKE exchanges conform to standard ISAKMP payload syntax and attribute encoding. The SA payload is always first in phase 1 messages. The hash payload must always be first in all phase 2 messages. Other ISAKMP payloads can be received in any order.

IKE Security Association Negotiation

The SA payload used by Main, Aggressive, and Quick Modes takes the form of *Transform* payloads encapsulated in *Proposal* payloads encapsulated in an SA payload. In phase 1 there can be only one SA payload and one Proposal payload. Different options for the ISAKMP SA are described in separate Transform payloads. In phase 2 it is possible to negotiate more than one SA at once. Hence there can be more than one SA payload. Each SA payload can have more than one Proposal payload (i.e. AH and ESP). Each Proposal payload can have more than one Transform payload (i.e. SHA or MD5).

IKE Key Exchange

IKE takes its key exchange method from RFC 2412, *The OAKLEY Key Determination Protocol*. It uses the Diffie-Hellman public key algorithm and can be completed in three or more messages. It provides the authenticated key exchange required by ISAKMP and can optionally provide PFS for phase 2. The Oakley protocol defines a set of Diffie-Hellman public key values known as the Oakley groups. IKE requires that the Oakley MODP Group 1 be supported in a compliant implementation.

IKE Authentication

IKE implementations must support authentication via pre-shared keys and may optionally support authentication with public key encryption, signatures, or a revised method of using public key encryption.

Authentication using a pre-shared key has the least computational overhead of the different authentication methods. However, the shared key must be securely transported to each device before ISAKMP negotiations take place. The key exchange is authenticated by adding the shared key to the key generation process. Each party then proves they know the value of the session key by encrypting and decrypting the last message in the phase 1 exchange.

Authentication using public key encryption is slower than pre-shared key but is a stronger method. The nonce payloads are encrypted using the peer's public key. The ability to decrypt the nonce authenticates the exchange. Public key encryption also provides phase 1 identity protection by encrypting the ID payload with the peer's public key.

Authentication using signatures is slower than pre-shared key but is a stronger method. Signed hashes are exchanged at the end of the exchange and authentication is achieved by verifying the signature with the peer's public key. Because only the owner of the private part of the public key pair can sign the hash, as long as the signature decrypts correctly the peer is the true owner of the public key.

Both public key encryption and signatures require that public keys for each party be exchanged and verified. In the simple case the public key can be exchanged manually and verified by the administrator. Alternatively the public keys can be extracted from a certificate that links the key to the peer and is signed by a trusted third party. IKE provides mechanisms for exchanging certificates by allowing messages to have an optional certificate payload or certificate request payload. This requires each IKE peer to have a separate method for obtaining certificates signed by a Certification Authority (CA).

Extended Authentication (XAUTH) allows legacy authentication mechanisms such as RADIUS to be used to extend phase 1 authentication. It does not replace phase 1 authentication and is completed in phase 1.5 before phase 2 SAs can be created.

IPsec on the Router

The router provides an RFC-compliant implementation of IPsec. The IPsec implementation supports AH and ESP, and integrates IPComp into IPsec for ease of management. IPsec is specified by the following RFCs:

- RFC 2104, "HMAC: Keyed-Hashing for Message Authentication".
- RFC 2401, "Security Architecture for the Internet Protocol".
- RFC 2402, "IP Authentication Header".
- RFC 2403, "The Use of HMAC-MD5-96 within ESP and AH".
- RFC 2404, "The Use of HMAC-SHA-1-96 within ESP and AH".
- RFC 2405, "The ESP DES-CBC Cipher Algorithm With Explicit IV".
- RFC 2406, "IP Encapsulating Security Payload (ESP)".
- RFC 2407, "The Internet IP Security Domain of Interpretation for ISAKMP".
- RFC 2408, "Internet Security Association and Key Management Protocol (ISAKMP)".
- RFC 2409, "The Internet Key Exchange (IKE)".
- RFC 2410, "The NULL Encryption Algorithm and Its Use With IPsec".
- RFC 2411, "IP Security Document Roadmap".
- RFC 2412, "The OAKLEY Key Determination Protocol".

IPsec is also compliant with the following Internet Drafts

- "The ISAKMP Configuration Method" Revision 5, 17 August 1999
- "Extended Authentication within ISAKMP/Oakley (XAUTH)" Revision 6, 1 December 1999
- "A Hybrid Authentication Mode for IKE" Revision 2, 21 June 1999

For backwards compatibility, the router also supports an older Security Association protocol based on 1998 Internet drafts and implemented in Software Release 7.7.4.

Modification of the IPsec configuration requires Security Officer privilege. See "Privilege Levels" on page 1-9 of Chapter 1, Operation and "The User Authentication Database" on page 1-16 of Chapter 1, Operation for more information about creating users and login names with Security Officer privilege.

IPsec uses ENCO services for various encryption, authentication, and compression algorithms. If the ENCO module is not configured to provide the resources required by IPsec, the IPsec configuration does not work as intended. See Chapter 25, Compression and Encryption Services for more information about configuring ENCO resources.

The router implements IPsec with the following components:

- The IPsec configuration, which is stored in the IPsec Security Policy Database (SPD). Three entities are used to define the IPsec configuration—SA specifications, bundle specifications, and policies. See "Security Policy Database (SPD)" on page 45-13 for a description of these entities.
- The currently active Security Associations and SA bundles. These are created from the IPsec configuration, either manually or by the ISAKMP/

IKE key management protocol. See "SA Bundles" on page 45-16 for a description of SAs and SA bundles. Manual and ISAKMP/IKE key management is described in "Security through Key Management" on page 45-16.

The IPsec implementation also provides backwards compatibility with the older Security Association implementation and a mechanism for converting old SA configurations to the equivalent IPsec configuration. "Pre-IPsec Security Associations" on page 45-20 describes this functionality.

Security Policy Database (SPD)

The protection IPsec offers is based on requirements defined by IPsec policies stored in a Security Policy Database (SPD) established and maintained by a *Security Officer*. When the router is in security mode, only users with Security Officer privilege can change the IPsec configuration. The three entities that define the IPsec configuration in the SPD are:

- **SA Specifications** are templates for SAs. They specify attributes an SA has when it is created.
- **Bundle Specifications** are templates for SA bundles. They specify the number and order of SAs an SA bundle has when it is created. A bundle specification used by ISAKMP/IKE to negotiate an SA bundle can specify more than one bundle of SAs, in order, with the most desired bundle first. ISAKMP/IKE negotiates with the IPsec peer to determine the bundle to use.
- **Policies** link together a rule for selecting a set of IP packets and an action. The actions are to permit, deny, or apply IPsec processing.

SA Specifications

The router creates Security Associations (SA) in pairs: one for outbound packets and one for inbound packets. When an SA pair is created, a specification determines SA attributes. Attributes of an SA specification are the key management mechanism to create SAs, encryption and/or authentication algorithms that SAs use, and how the SAs provide an anti-replay service, if any. When the specification requires a manual key management mechanism to create SAs, the specification identifies SPIs and encryption and/or authentication keys that the SAs need.

Bundle Specifications

The router creates an SA pair as part of an SA bundle. However, there is often only one pair in a bundle. When a bundle is created, a specification determines its attributes. Attributes of a specification is the key management mechanism to create the bundle, the SA pairs created for the bundle, and the lifetime expiry limits of the SA pairs created for the bundle. SA pairs are represented by SA specification identification numbers in a *bundle string*, which can also contain the connectors *and*, *or*, and a comma.

Bundle specifications that specify manual key management to create a bundle can specify that it consist of one, two, or three SA pairs. Each pair must use a different IPsec protocol (ESP, AH, or IPComp.) The SA specification identification numbers in the bundle string that create the pairs are separated by *ands*. SAs are applied to outbound packets in the order of the respective SA specifications in the bundle string. They are applied in the reverse order to inbound packets.

Bundle specifications that specify ISAKMP/IKE to create the bundle can specify one or more proposals, each consisting of one, two, or three SA pairs. Each of these proposals is separated by a comma in the bundle string. Pairs proposed in the bundle string can have more than one SA specification proposed for it. If different SA specifications offer a choice of algorithm(s) for the same IPsec protocol, they are separated by *or* in the bundle string.

For example, assume the following three SA specifications have been created:

- SA specification 1: ESP providing triple DES encryption.
- SA specification 2: ESP providing DES encryption.
- SA specification 3: AH providing MD5 authentication.

A valid bundle string would be:

```
"1 OR 2 AND 3, 1 OR 2"
```

This bundle string specifies two proposals. The preferred proposal, which is listed first, ("`1 OR 2 AND 3`") proposes a bundle of two SA pairs; an ESP SA pair ("`1 OR 2`") and an AH SA pair ("`3`"). There is a choice for the ESP SA pair between triple DES ("`1`") and DES ("`2`") with triple DES being the preferred choice as it is listed first. The AH SA pair must use MD5.

The second proposal ("`1 OR 2`") proposes a bundle of one SA pair; an ESP SA pair with a choice between triple DES ("`1`") and DES ("`2`") with triple DES being the preferred choice as it is listed first.

Other possible bundle strings using these SA specifications include:

```
"1 AND 3, 1, 2 AND 3, 2"

"1, 2 AND 3"

"1, 2"
```

IPsec Policies

In general, the mode that processes each IP packet is determined by matching information from the IP and transport layer headers of the IP packet with IPsec policies in the SPD.

An IPsec policy binds a packet selection rule to an action. A policy can be attached to only one IP logical interface, but multiple IPsec policies can be attached to the same IP logical interface. When multiple policies are attached to one IP logical interface, the policies are ordered and packets traversing the interface are matched against the policies' selection rules in order. Policies with more specific selection rules must be placed ahead of those with general rules. The commands **create ipsec policy** command on page 45-48 and **set ipsec policy** command on page 45-75 allow the order of policies on an interface to be managed. IPsec polices are attached to IP logical interfaces and receive packets for processing when the IPsec module is enabled.

Information used to select packets are called *selectors*. The packet selection rule is defined in a policy by assigning a value to one of the following selectors:

- Remote IP address – a single address, a range of addresses or a masked subnet
- Local IP address – a single address, a range of addresses or a masked subnet
- Remote port

- Local port
- Transport protocol such as UDP, TCP, and ICMP
- Local system name or user name
- Remote system name or user name

If a value is not assigned to a selector, the default is to match any value. An IP packet must have fields that match all the selectors of a policy to be selected by a policy. Each SA pair in each SA bundle associated with a policy also has selectors and selection values assigned to them.

A policy can have one SA bundle that processes all packets for a policy (in this case the pairs in the bundle have the same selection values as the policy). Or a policy can have more than one bundle, each of which processes a sub-set of the packets that match the policy. In this case, the pairs in each bundle have selection values that are more specific than those on the policy – their selection values for one or more selectors are the actual values seen in IP packets matching the policy. The configuration of the policy determines which of these two cases will exist. Each policy has a parameter (**saselectorfrompkt**) that specifies the SA selectors that take their selection value from the associated field of the IP packet rather than from the policy selection.

An IPsec policy can define these actions:

- **permit** – allow matching packets to bypass IPsec and be processed normally by the router
- **deny** – discard matching packets
- **ipsec** – apply IPsec processing to matching packets

If the action is **ipsec**, the policy must also specify:

- The IP address of the IPsec peer, or **any** or **dynamic**. See "Dynamic IP Addresses" on page 45-17.
- The key management method to be used for creating an SA bundle to process the selected packets (either **isakmp** or **manual**).
- The bundle specification to be used when creating an SA bundle to process the selected packets.

If the key management is ISAKMP, the policy may also specify:

- That ISAKMP/IKE must use Perfect Forward Secrecy (PFS) when creating keys for the bundle.
- An Oakley group to be used by ISAKMP/IKE when negotiating the bundle.
- That Extended Authentication (XAUTH) is to be used. If the router is to be used as an XAUTH client, then an XAUTH username and password must also be supplied. Hybrid XAUTH allows phase 1 client authentication to be skipped but can only be used with an authentication type of **rsasignatures**.

SA Bundles

Each SA pair created on the router is stored in the Security Association Database (SAD) on the router. Each pair belongs to one SA bundle. Each bundle created by the router is attached to one IPsec policy and more than one bundle can be attached to the same policy. When multiple bundles are attached to a policy, the first one with pairs whose selectors match the packet is used to process the packet.

Multiple bundles may be assigned to one policy because:

- A second bundle may be created to replace a bundle that is nearing its lifetime expiry limit. In this case, there are two bundles that differ only in the keys and SPIs of their SA pairs. The new bundle is always placed before the old one and is therefore chosen to process all packets.
- A policy may be configured so that one or more of its selectors creates a new bundle for each new packet value seen of that selector. In this case, multiple bundles are active at any one time, and the bundle used to process a packet is the one with selectors that match the packet.

Create an SA bundle in one of the following ways:

- Manually, in which case the SA specifications used to create the SA pairs in the bundle specify the keys and SPIs to use.
- Using ISAKMP/IKE, in which case ISKAMP/IKE negotiates the keys and SPIs to use for the SA pairs.

Each SA pair is attached to one or more ENCO channels. One channel is for encryption and decryption when the pair provides encryption; one channel is for authentication when the pair provides authentication; and one channel is for compression and decompression when the pair provides compression.

Security through Key Management

IPsec security services use cryptographic keys for authentication and encryption. IPsec relies on a separate set of mechanisms to put these keys in place. IPsec requires support for both manual and automatic distribution of keys. ISAKMP/IKE is the default automatic key management mechanism. Other automated key distribution techniques may be used. For more information, see "ISAKMP/IKE" on page 45-6.

Manual Key Management

When an IPsec policy is configured to use manual key management to create its SA bundles, all bundle specifications and SA specifications that the policy uses must have their key management attribute set to **manual**.

When using manual key management, SA bundles are created when:

- Policies are created using the **create ipsec policy** command on page 45-48, and IPsec has previously been enabled with the **enable ipsec** command on page 45-66.
- IPsec is enabled with the **enable ipsec** command on page 45-66, and policies have previously been created using the **create ipsec policy** command on page 45-48.
- Parameters of the policy that affect the IPsec processing configuration are changed with the **set ipsec policy** command on page 45-75, and IPsec has

previously been enabled with the **enable ipsec** command on page 45-66. In this case, the current SA bundles are destroyed first.

When using manual key management, SA bundles are destroyed when:

- An SA bundle or a policy is explicitly destroyed using the **destroy ipsec policy** command on page 45-60, and IPsec has previously been enabled with the **enable ipsec** command on page 45-66.
- IPsec is disabled using the **disable ipsec** command on page 45-62.
- Parameters of the policy affecting the IPsec processing configuration are changed with the **set ipsec policy** command on page 45-75, and IPsec has previously been enabled with the **enable ipsec** command on page 45-66. In this case, new SA bundles are created after the current ones are destroyed.

ISAKMP/IKE Key Management

When an IPsec policy is configured to use ISAKMP/IKE to create its SA bundles, all bundle specifications and SA specifications that the policy uses must have their key management attribute set to **isakmp**.

When using ISAKMP/IKE key management, IPsec requests that a new bundle be negotiated when the IP module passes it a packet for a policy that does not have a bundle or does not have a bundle with selectors that match the packet. Outbound packets are queued while IPsec is waiting for ISAKMP/IKE to negotiate a bundle. SA bundles are also created when an IPsec peer requests that ISAKMP/IKE negotiates a bundle.

SA pairs negotiated by ISAKMP/IKE have lifetime expiry limits, specified in seconds and/or kilobytes of data processed. When a pair in a bundle nears its lifetime limit, IPsec requests that ISAKMP/IKE negotiate a replacement bundle. The replacement is used as soon as it is negotiated.

Dynamic IP Addresses

In the simplest case two IPsec peers have fixed IP addresses. The IPsec and ISAKMP policies they use to communicate can be configured with these addresses. However, it is possible for a router with a dynamically assigned IP address to protect IP traffic using IPsec, provided that it uses ISAKMP key management and only communicates with IPsec peers that have fixed IP addresses.

Whenever an IPsec peer with a fixed IP address (Router A) is communicating with an IPsec peer with a dynamically assigned IP address (Router B) it needs to know the system name or user name of Router B so that it can select the correct IPsec and ISAKMP policies to use for communications with Router B. If it can also verify a password associated with Router B's name the name can be used to verify that the IPsec peer is indeed Router B.

Router A and Router B can use one of the following methods for their IPsec communications:

- Router A has an ISAKMP policy specifying **any** peer. Router B has an ISAKMP policy specifying the IP address of Router A.

  Router A has an IPsec policy configured with **peeraddress**=**dynamic** and **rname** set to the name of Router B. Router B has an IPsec policy with **peeraddress** set to the IP address of Router A and **lname** set to its own name.

- Router A has an ISAKMP policy specifying **any** peer and XAUTH configured. Router B has an ISAKMP policy specifying the IP address of Router A and with XAUTH configured with its own user name and password.

  Router A has an IPsec policy configured with **peeraddress=any**. Router B has an IPsec policy with **peeraddress** set to the IP address of Router A.

  In this case, the same IPsec policy is used for any peer. Authentication of the peer is provided by ISAKMP XAUTH.

In both of the above cases it may be necessary for Router A to add an IP route for the network(s) represented by Router B when it discovers the actual IP address that Router B is using. This is accomplished by specifying an IP route template on Router A's IPsec policy. When the policy is used to establish communications with a peer, an IP route is added using the information in the specified route template. [For information on setting up IP route templates see "Route Templates" on page 14-29 of Chapter 14, Internet Protocol (IP).]

IPsec Support for IPv6

IPsec is mandatory in IPv6. With encryption key handling (see Chapter 25, Compression and Encryption Services), it provides authentication and encryption to all IPv6 traffic.

To create IPsec policies for IPv6 traffic, use the command:

```
create ipsec policy=name interface=interface action={deny|
    ipsec|permit} ipversion=6
    [bundlespecification=bundlespecification-id] [group={0|1|
    2}] [icmptype={list|ndall}] [isakmppolicy=isakmp-policy-
    name] [keymanagement={isakmp|manual}] [laddress={any|
    ipv6add[/prefix-length]|ipv6add-ipv6add}] [lname={any|
    system-name}] [lport={any|opaque|port}]
    [peeraddress={ipv6add|any|dynamic}] [position=pos]
    [raddress={any|ipv6add[/prefix-length]|ipv6add-ipv6add}]
    [rname={any|system-name}] [rport={any|port|opaque}]
    [saselectorfrompkt={all|laddress|lport|none|raddress|
    rport|transportprotocol}] [trasnportprotocol={any|egp|
    esp|gre|icmp|opaque|ospf|rsvp|tcp|udp|protocol}]
    [usepfskey={true|false}]
```

Setting the **ipversion** parameter to 6 indicates that this policy applies to IPv6 traffic and that only IPv6 addresses and network connections are valid.

The **icmptype** parameter includes an **ndall** option, which is equivalent to specifying types 133,134,135 and 136 (the types that are required for IPv6 neighbour discovery). IPv6 routing functions correctly when these packet types are accepted and processed by the router.

To modify existing IPsec policies, use the command:

```
set ipsec policy=name [other options]
```

To create ISAKMP policies for key management for IPv6 traffic, use the command:

```
destroy isakmp policy=name peer={ipv6add|any} ipversion=6
    [authtype={preshared|rsaencr|rsasig}]
    [dhexponentlength=160..1023] [encalg={3des2key|3desinner|
    3desouter|des}] [expirykbytes=1..1000]
    [expiryseconds=600..31449600] [group={0|1|2}]
```

```
[hashalg={SHA|MD5}] [key=0..65535] [localid={ipv6add|
domainname|user-domainname|dist-name}]
[localrsakey=0..65535] [mode={main|aggressive}]
[msgretrylimit=0..1024] [msgtimeout=1..86400]
[phase2xchglimit={none|1..1024}]
[policyfilename=filename] [prenegotiate={on|off|true|
false}] [senddeletes={on|off|true|false}] [sendnotify={on|
off|true|false}] [sendidalways={on|off|true|false}]
[setcommitbit={on|off|true|false}] [remoteid={ipv6add|
domainname|user-domainname|dist-name}]
```

Connections can be accepted from any IPv6 address by setting the **peer** parameter to **any**.

To modify existing ISAKMP policies, use the command:

```
set isakmp policy=name [other options]
```

IPsec over UDP

Intermediate devices that modify the IP header often prevent IPsec from operating correctly. In general, devices fail to operate correctly or block the packet when they require more information about the packet than is in the IP header. These types of devices function effectively with UDP and TCP, but do not understand the data streams created by IPsec (AH, ESP, and IPComp), and typically can only handle one IPsec flow from one source to one destination.

To solve this problem, IPsec over UDP encapsulates IPsec inside UDP packets. Packets are encrypted and authenticated with ESP, which authenticates the packet payload only. AH cannot be made to function with devices that modify the IP header. The AH protocol is unable to authenticate this changed header.

To enable UDP tunnelling for IPsec packets on a per policy basis, use the command:

```
set ipsec policy=name udptunnel=true
```

When tunnelling is enabled for the router policy, the router encapsulates traffic processed by IPsec inside UDP packets. The remote device decapsulates the packets and processes them normally. Therefore, intermediate devices are needed only to process UDP packets.

Many gateway devices keep a state for UDP data streams that expires unless refreshed frequently. To prevent this expiry, UDP heartbeats can be enabled on a per-policy basis. Heartbeat packets are sent that periodically refresh the state entries of intermediate devices. To enable UDP heartbeats, use the command:

```
set ipsec policy=name udpheartbeats=true
```

By default, the router listens on and sends all IPsec packets over UDP port 2746. The listen port is created when the first IPsec policy with UDP tunnelling enabled is created, and is closed when the last policy with UDP tunnelling enabled is destroyed.

Both the local listen and the remote destination ports can be changed from the default. To change the remote port to which the UDP tunnelled packets are sent, use the command:

```
set ipsec policy=name edpport=port
```

To change the local listen port, use the command:

```
set ipsec udpport=port
```

IPsec over UDP packets, being a generic UDP data stream, are blocked by the router unless a permit policy is created that allows them to pass through. To create this permit policy, use the command:

```
create ipsec policy=name action=permit lport=2746
```

In this example, the local listen port is the default (2746). This policy must come before other policies on an IPsec enabled interface.

Pre-IPsec Security Associations

The router supports, for backward compatibility, the Security Association implementation, based on 1998 Internet Drafts, as implemented in Software Release 7.4.

The software allows mixed networks of old Security Association implementations and IPsec. An IP logical interface may have both IPsec policies and old SAs attached to it. This allows a router to communicate using old SAs with one set of routers and to communicate using IPsec with another set of routers. To enable backward compatibility, use the command:

```
enable ipsec oldsa interface=interface
```

To upgrade a router from the old SA implementation to IPsec, use the command:

```
activate ipsec convertoldsa [sa=sa-id]
```

to convert the old SA configuration in memory into a functionally equivalent IPsec configuration in memory. The old SA configuration is removed from memory. After the conversion process, the IPsec policies created do not have valid IPsec peer addresses. These need to be entered manually before the new IPsec configuration can be used. This conversion process does not change the router configuration file in any way. To retain the new IPsec configuration over a router restart, update the configuration file with the command:

```
create config=filename
```

Alternatively, edit the current boot configuration script with the command:

```
edit filename
```

ISAKMP/IKE on the Router

ISAKMP/IKE is implemented with ISAKMP commands, and changing the ISAKMP configuration requires Security Officer privilege.

ISAKMP negotiates security associations on behalf of IPsec. When an IPsec policy specifies ISAKMP for key management, ISAKMP must be enabled and an ISAKMP policy must exist for the ISAKMP peer.

To enable or disable ISAKMP, use the commands:

```
enable isakmp [localrsakey=key-id] [policyserverenabled={on|
   off|true|false}] [policyfilename=filename] [udpport=port]

disable isakmp
```

To display the status of ISAKMP, use the command:

```
show isakmp
```

The router uses ENCO services for encryption, authentication, and key storage services. See Chapter 25, Compression and Encryption Services for more information about configuring ENCO resources. ISAKMP requires DES encryption and the Diffie-Hellman key exchange algorithm.

Because ISAKMP uses UDP port 500 as its transport protocol, care must be taken to ensure this traffic is not blocked by IP or IPsec. Typically, an IPsec **permit** policy must be added to the interface over which ISAKMP traffic is sent and received. An example of a command to add a **permit** policy for ISAKMP traffic is:

```
create ipsec policy=isakmp interface=ppp0 action=permit
   lport=500 rport=500
```

ISAKMP Policies

An ISAKMP policy specifies how to communicate with an ISAKMP peer and how to authenticate one. The information in the ISAKMP policy is used during ISAKMP exchanges and when an ISAKMP SA is created.

An ISAKMP policy specifies an encryption algorithm and a hash algorithm. The encryption algorithm encrypts ISAKMP messages to protect them against eavesdropping. The hash algorithm authenticates ISAKMP messages to prevent man-in-the-middle attacks.

An ISAKMP policy must also specify the address of the remote ISAKMP peer. This is used to match an ISAKMP policy to an IPsec peer. When IPsec requests the negotiation of an IPsec SA, it supplies both the IPsec SA details and the IP address of the ISAKMP peer. The IP address is used to find the ISAKMP policy with the matching peer address, and this policy is used for the negotiation. If the policy peer address is set to **any,** the policy is used to match a remote ISAKMP peer with an unknown IP address.

An ISAKMP policy must specify one of the following methods to authenticate an ISAKMP peer:

- Pre-shared key
- RSA encryption
- RSA signatures

A pre-shared key requires that a secret key be transported to both ISAKMP peers securely before the ISAKMP negotiation takes place.

RSA encryption and RSA signatures both require that RSA public keys be exchanged before ISAKMP negotiations take place. However, this can be done using methods that are not secure because RSA public keys are not secret. See Chapter 25, Compression and Encryption Services for more information about transferring ENCO keys between routers.

An ISAKMP policy may also specify a different Diffie-Hellman group from the Oakley MODP Group 1 default. The more secure Oakley MODP Group 2 group may be used at the expense of negotiation speed. A less secure MODP Group 0 is also available.

On AR440S, AR441S and AR450S routers only, you can specify the *Advanced Encryption Standard* (AES) algorithm for ISAKMP policies and SA specifications. AES is a FIPS approved symmetric block cipher that is documented in FIPS PUB 197. AES supports variable key lengths that can be longer than DES (and Triple DES) keys. With AES you specify a 128, 192 or 256 bit key that is used to generate sub-keys. These sub-keys and 128 bit blocks of data are then subjected to the encryption and decryption routines.

A feature licence is required to use AES encryption. Contact your authorised distributor or resellerfor details.

To create, modify, or destroy an ISAKMP policy, use the commands:

```
create isakmp policy=name peer={ipadd|any}
    [authtype={preshared|rsaencr|rsasig}]
    [dhexponentlength=160..1023] [encalg={3des2key|
    3desinner|3desouter|des|aes128|aes192|aes256]
    [expirykbytes=1..1000] [expiryseconds=600..31449600]
    [group={0|1|2}] [hashalg={sha|md5}] [hybridxauth={on|off|
    true|false}] [key=0..65535] [localid={ipadd|domainname|
    user-domainname|dist-name}] [localrsakey=0..65535]
    [mode={main|aggressive}] [msgretrylimit=0..1024]
    [msgtimeout=1..86400] [phase2xchglimit=none|1..1024]
    [policyfilename=filename] [prenegotiate={on|off|true|
    false}] [remoteid={ipadd|domainname|user-domainname|
    dist-name}] [senddeletes={on|off|true|false}]
    [sendnotify={on|off|true|false}] [sendidalways={on|off|
    true|false}] [setcommitbit={on|off|true|false}]
    [srcinterface=interface] [xauth={client|server|none}]
    [xauthname=username] [xauthpasswd=password]
    [xauthtype={generic|radius}]
```

```
set isakmp policy=name [authtype={preshared|rsaencr|rsasig}]
    [dhexponentlength=160..1023] [encalg={3des2key|3desinner|
    3desouter|des|aes128|aes192|aes256}]
    [expirykbytes=1..1000] [expiryseconds=600..31449600]
    [group={0|1|2}] [hashalg={sha|md5}] [hybridxauth={on|off|
    true|false}] [key=0..65535] [localid={ipadd|domainname|
    user-domainname|dist-name}] [localrsakey=0..65535]
    [mode={main|aggressive}] [msgretrylimit=0..1024]
    [msgtimeout=1..86400] [phase2xchglimit={none|1..1024}]
    [polichfilename=filename] [prenegotiate={on|off|true|
    false}] [remoteid={ipadd|domainname|user-domainname|
    dist-name}] [senddeletes={on|off|true|false}]
    [sentnotify={on|off|true|false}] [sendidalways={on|off|
    true|false}] [setcommitbit={on|off|true|false}]
    [srcinterface=interface] [xauth={client|server|none}]
    [xauthname=username] [xauthpasswd=password]
    [xauthtype={generic|radius}]

destroy isakmp policy=name
```

To display the currently defined ISAKMP policies or details of a particular policy, use the command:

```
show isakmp policy[=name]
```

ISAKMP Exchanges

An ISAKMP SA must exist between two IPsec peers before IPsec SA negotiations can take place. If the ISAKMP policy specifies that the ISAKMP SA is to be pre-negotiated, then the ISAKMP SA is created on router startup. When no ISAKMP SA exists when IPsec requests that an SA be negotiated, the phase 2 exchange is queued and a phase 1 exchange starts. After the phase 1 exchange successfully creates the ISAKMP SA, phase 1.5 exchanges start. When phase 1.5 exchanges finish or none are required, phase 2 exchanges start.

To display details of currently active ISAKMP exchanges, use the command:

```
show isakmp exchange[=exchange-id]
```

ISAKMP Security Associations (SA)

The ISAKMP SA protects ISAKMP traffic between the local router and the remote ISAKMP peer. It ensures that SAs and keys negotiated for other security services such as IPsec are kept secret and are authenticated. It also provides identity protection for the SAs created. It is created during a phase 1 exchange using information stored in the ISAKMP policy. A single ISAKMP SA is dynamically created for each ISAKMP peer. Algorithms that encrypt and authenticate messages from the remote ISAKMP peer are stored in the ISAKMP SA along with keys for the algorithms. The ENCO Diffie-Hellman key exchange algorithm creates keys for ISAKMP SAs.

The ISAKMP SA is identified by cookies in the ISAKMP message header and is stored locally in an ISAKMP SA database. Different SA definitions are defined in a separate document called the Domain of Interpretation (DOI). The DOI provides an interpretation of DOI-specific payloads as well as defining protocol and algorithm identifiers.

To display details of existing ISAKMP SAs, use the command:

```
show isakmp sa[=sa-id]
```

ISAKMP Heartbeats

ISAKMP heartbeat exchanges provide additional security by enabling the central router to detect when the connection is lost between it and the remote router. The dial-up router sends a heartbeat message to the central router every 20 seconds. If three messages in a row are not received, the central router concludes that the line has been lost and deletes the ISAKMP SAs for the remote router.

To enable or disable ISAKMP heartbeat exchange for an existing policy, use the command:

```
set isakmp policy=name heartbeatmode={noth|none|receive|send}
```

To enable ISAKMP heartbeat exchange for a new policy, use the command:

```
create isakmp policy=name heartbeatmode={bith|none|receive|
    send} [other parameters]
```

The remote router should be set to **heartbeatmode=send** and the central router to **heartbeatmode=receive**.

To display the current heartbeat mode, use the commands:

```
show isakmp policy=name
```

```
show isakmp sa=sa-id
```

To display the number of heartbeat packets received, use the command:

```
show isakmp counters
```

IPsec NAT-Traversal

IPsec NAT-Traversal is an enhancement to IPsec and ISAKMP protocols that lets Virtual Private Network (VPN) clients communicate through NAT gateways over the Internet. For example, business travellers commonly use IPsec on their laptops to gain remote VPN access to the central office. When working off-site, these users sometimes need to connect to the Internet through a NAT gateway such as from a hotel. Network Address Translation (NAT) gateways are often part of a company's firewall and let its Local Area Network (LAN) appear as one IP address to the world. For more information about NAT devices, refer to RFC 1631 and to "Network Address Translation" on page 14-48 of Chapter 14, Internet Protocol (IP).

Problems arise with NAT devices for a number of reasons. A key one is that when they handle IPsec packets, they cannot access encrypted UDP or TCP headers. Therefore, NAT devices cannot identify traffic for different private devices and cannot properly track individual sessions.

NAT-T is not on the NAT gateway and is not an "IPsec pass-through". NAT-T lets IPsec/ISAKMP peers send traffic through NAT gateways by putting packets inside UDP packets. This solution enables remote VPN users to communicate successfully when NAT gateways are part of the connection.

Basic NAT-T Operations

Using NAT-D (discovery) messages, NAT-T negotiates with a peer to determine if NAT gateways are present and at which end of the network. Each peer sends at least two NAT-D messages as part of the ISAKMP phase 1 negotiation. The first message contains a hash of a destination IP address; subsequent messages contain source addresses. A NAT device is detected when address messages from the peer have incorrect hash values, which indicates that a NAT device changed IP addresses.

Also during phase 1, NAT-T determines whether a peer has NAT-T capabilities by detecting a vendor ID. Vendor IDs tell what version of NAT-T the peer supports. When a NAT device is not detected or a peer does not support NAT-T, normal IPsec negotiations and protection occur.

When an ISAKMP initiator detects a NAT device during an exchange, communication changes from UDP port 500 to port 4500. Log messages inform users that the UDP port has changed. Main or Aggressive mode packets received on the old port are discarded and a separate log is created.

Because IPsec traffic can also be received on port 4500, ISAKMP adds and removes the non-ESP marker at the start of the ISAKMP message so that messages can be detected and passed to the ISAKMP module. ISAKMP drops packets when it receives them on port 4500 without a non-ESP marker.

NAT-T inserts a UDP header between the outer IP and ESP headers thereby encapsulating the ESP data (figure below). NAT-T encapsulates IPsec traffic only when a NAT device is detected.

Figure 45-1: UDP Encapsulation for NAT-T

IP Header	**New UDP Header**	ESP Header	Encrypted Data

IPsec intercepts UDP-encapsulated ESP packets before they are passed to UDP.

A peer behind a NAT device sends keepalive messages to ensure that port mappings in the device remain active between peers. Keepalive intervals are not configurable. The purpose of keepalive messages is different from heartbeat messages controlled by the **heartbeatmode** parameter in ISAKMP policy commands, which detect an IKE peer. IKE heartbeat messages and NAT-T keepalive messages do not affect each other.

NAT-T on the Router

This NAT-T implementation supports interoperability with the following VPN clients:

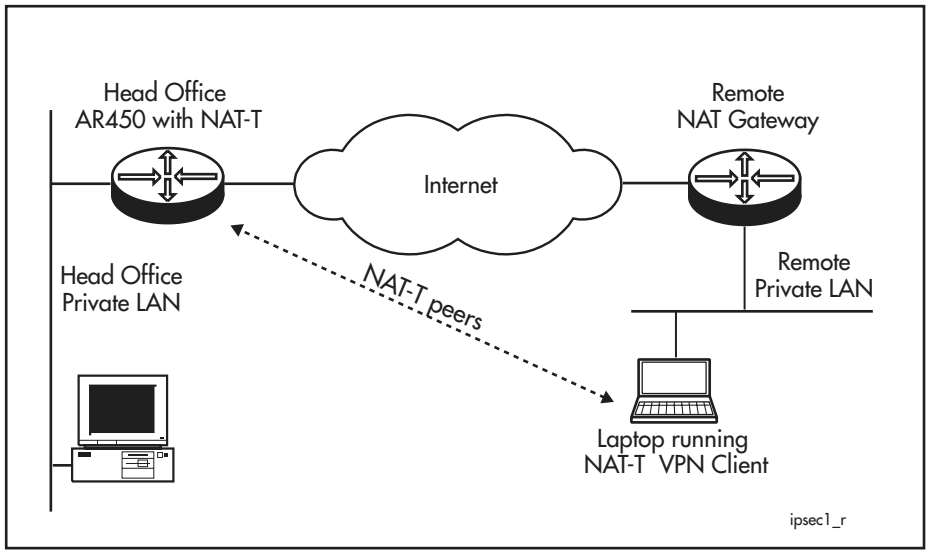
- Microsoft Windows 2000®
- Microsoft Windows XP®
- SafeNet SoftRemote®

NAT-T can also be implemented router-to-router for offices with their own IPsec router behind a NAT device.

NAT-T is compatible with the following IETF Internet-Drafts:

- draft-ietf-ipsec-nat-t-ike-02, *Negotiation of NAT-Traversal in the IKE*, which describes the modifications to IKE to support NAT detection and UDP tunnel negotiation
- draft-ietf-ipsec-udp-encaps-02, *UDP Encapsulation of IPsec Packets*, which defines the method of UDP encapsulation of IPSec packets
- draft-ietf-ipsec-nat-t-ike-08, *Negotiation of NAT-Traversal in the IKE*, which describes the modifications to IKE to support NAT detection and UDP tunnel negotiation
- draft-ietf-ipsec-udp-encaps-08, *UDP Encapsulation of IPsec Packets*, which defines the method of UDP encapsulation of IPSec packets

Figure 45-2: NAT-T peers negotiate traffic through a NAT gateway device



NAT-T is enabled by default, and is enabled or disabled in the ISAKMP policy. We recommend that users carefully read security considerations in the IETF drafts to fully understand the implications of using NAT-T. When users create

an ISAKMP policy with the **create isakmp policy** command, they define how peers respond during an ISAKMP exchange, and can use the **nattraversal** parameter to disable NAT-T if they prefer. They can later use the same parameter with the **set isakmp policy** command to enable NAT-T again.

Users should configure an IPsec policy to allow ISAKMP traffic on UDP port 4500 so that it flows through the IPsec layer to ISAKMP. Refer to the ISAKMP policy commands in this chapter to change or add a policy.

Peers send their original, untranslated addresses to each other, which they store in the ISAKMP SA. Recipients use original addresses (OAs) to correct the checksums in the UDP or TCP headers in the IPsec payload.

NAT-T is implemented for IPv4 for both transport and tunnel modes. We recommend transport mode for MS clients. For SafeNet and router-to-router connections, we recommend tunnel mode and specifying unique IP addresses for remote peers. Refer to the **set ipsec saspecification** command to set modes.

Pre-IPsec Security Associations

An older implementation of Security Associations (based on RFCs 1825, 1827, and 1829 and subsequently updated by Internet Drafts) provided IP payload encryption using a method for encapsulating DES-encrypted IP payloads in IP packets.

IPsec outdates the functionality of the old Security Association (SA) implementation, but we still support this SA implementation for backward compatibility. There are easy upgrade options to convert an old SA configuration to IPsec. For more information on updating an old SA configuration to IPsec, see "Pre-IPsec Security Associations" on page 45-20.

The old SA implementation uses ENCO services to provide its own set of services to IP. IP uses SAs to implement IP payload encryption. AT-VPNet uses security associations and IP payload encryption to create secure virtual private networks across the Internet.

An SA defines a security transform to be applied to all traffic between any of its members. A security association exists on one or more routers and defines a Virtual Private Network (VPN).

A security association is identified by its Security Parameters Index (SPI). The SPI must be same on all instances of the security association throughout the VPN.

The members of an IP security association are IP addresses or contiguous groups of addresses. A member is defined by a base IP address and a network mask. A member is local to a given router if it is separated from the Internet by the router, otherwise it is remote. Thus all members of a security association are local to one router and remote to the other routers in the VPN.

An IP interface may have zero or more security associations. If an IP interface does not have an assigned security association, all IP packets transiting the interface are processed normally by the IP routing software. If an IP interface uses one or more security associations it passes all packets transiting the interface to the SA. The SA examines the source and destination addresses of a packet to determine if the packet is in any of the security associations used by the interface. If a match is found the packet is forwarded to the ENCO module on the channel to which the security association is attached. In this way the configured transform is applied to the packet. If a packet is not in one of the security associations it is discarded or passed through the interface, depending on the setting of the **samode** parameter for the IP interface.

Currently the security transforms available are DES CBC, 2-key Triple DES, and 3-key Triple DES Inner mode.

The SA module attaches to ENCO channels. The ENCO channels are configured not to retain process histories between data packets as the IP module does not require that packets be decrypted in the same order they were encrypted. Re-ordered or lost IP packets do not cause subsequent packets to be discarded due to decryption check errors and therefore the SA module never needs to reset its ENCO channels.

The SA module uses statically defined, manually managed keys. A network key has to be created on one router and then entered into all the other routers in the VPN. For more information about creating and entering keys into the router, see Chapter 25, Compression and Encryption Services.

Configuration Examples

The following examples illustrate how to configure IPsec and ISAKMP/IKE on a router.

- **VPN-only with details about ISAKMP/IKE key management**
- **VPN with NAT-Traversal**
- *How To Configure Microsoft Windows XP Virtual Private Network (VPN) client interoperability with NAT-T support* at www.alliedtelesyn.co.nz/solutions/solutions.html
- *How To Configure Microsoft Windows 2000 Virtual Private Network (VPN) client interoperability with NAT-T support* at www.alliedtelesyn.co.nz/solutions/solutions.html
- *How To Configure Microsoft Windows 2000 Virtual Private Network (VPN) client interoperability without NAT-T support* at www.alliedtelesyn.co.nz/solutions/solutions.html
- *How To Configure Microsoft Windows XP Virtual Private Network (VPN) client interoperability without NAT-T support* at www.alliedtelesyn.co.nz/solutions/solutions.html

A 3DES algorithm is used in the VPN-only example but may also be used in a configuration for VPN with NAT-Traversal. This means that a 3DES feature licence key must be on the router. Licence information can be obtained from your authorised distributor or reseller.

Some interface types, port types, or command options in these examples may not be supported on your router. Interfaces and port types vary depending on the router model, and whether an expansion unit (PIC, NSM) is installed. For more information, see the Hardware Reference.

While switches can be configured as VPN gateways, you should consider the performance implications. Throughput is affected on interfaces to which IPsec policies are attached. Therefore, these interfaces would lose wire-speed switching of data.

Setting Security

Before you begin, security keys must be created by entering a series of commands on a terminal connected directly to port 0 (or to the console port) on the router. The values for the keys must be the same on both routers.

It is important that your router be in security mode to avoid keys being destroyed when you restart it.

4. Define a security officer.

Complete this step on both the head office and remote site routers.

The following terminal output shows prompts along with commands you must enter (bold text) to define a security officer. You must define these on each router that uses the keys.

```
Manager> add user=secoff password=your-password
priv=securityofficer
User Authentication  Database
-----------------------------------------------------------
Username: secoff ()
Status: Enabled Privelige: Sec Off Telnet: No
Logins: 0 Fails: 0 Sent: 0 Rcvd: 0
-----------------------------------------------------------
Manager> enable system security
Info (134003): Operation successful.
Manager> login secoff
Password:
-----------------------------------------------------------
```

5. Generate a pre-shared key at the head office router.

For a router-to-router solution, generate a random key with the command:

```
create enco key=1 type=general random
```

Or to also allow incoming Microsoft VPN clients, enter the pre-shared key as alphanumeric with the command:

```
create enco key=1 type=general value=alphanumeric-key
```

To display the key value, use the command:

```
show enco key=1
```

Note the value so that you can load it on the remote router. This pre-shared key is used to encrypt ISAKMP negotiation. See "ISAKMP/IKE" on page 45-6 for more information.

See Chapter 25, Compression and Encryption Services for more information about generating and entering keys.

6. Create the same pre-shared key at the remote site.

To enter the random or alphanumeric key from the previous step, use the command:

```
create enco key=1 type=general value=secret-key
```

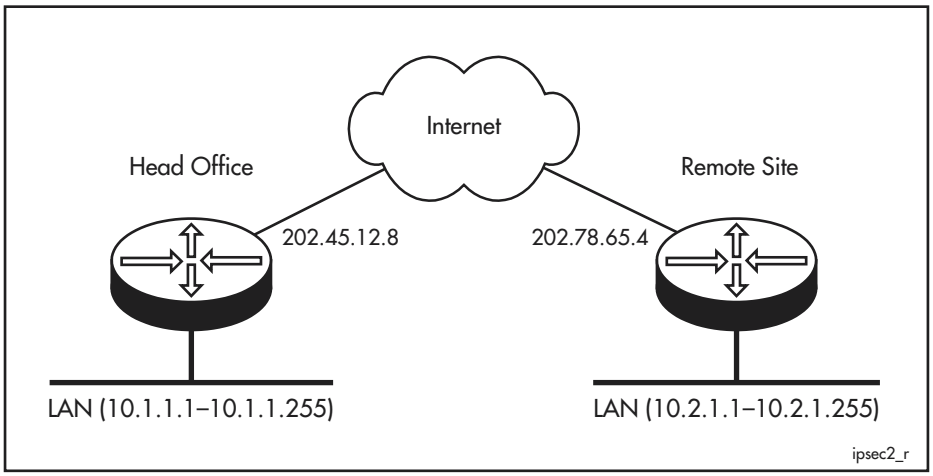
VPN-only with details about ISAKMP/IKE key management

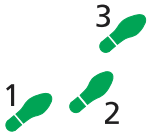
This example illustrates how to configure and use IPsec with ISAKMP key management for a VPN-only Internet connection. ISAKMP/IKE negotiates the required IPsec SAs between two routers and sets up the secret keys required by the encryption and authentication algorithms. The configuration sets up an encrypted VPN over the Internet that is for VPN only, and does not permit Internet browsing.

When ISAKMP/IKE starts negotiating IPsec protocols and secret keys between the two routers, the ISAKMP traffic is protected by an ISAKMP SA. The encryption algorithm and hash algorithm that protect ISAKMP traffic must be specified in the relevant ISAKMP policy. DES and SHA are used in the examples in this section.

In this example, the head office router has the resources to perform 3DES encryption, and is configured to use 3DES2KEY as the preferred encryption algorithm. However, the remote router does not have the resources or a licence to use 3DES encryption so is configured to perform DES encryption. Therefore, DES is chosen when ISAKMP/IKE negotiates the encryption algorithm between the two routers. But if the remote router can perform 3DES encryption later, the configuration on the head office router still works. Both routers prefer the SHA hash algorithm to MD5 or DESMAC.

Figure 45-3: Network for VPN-only with ISAKMP/IKE key management.





To configure the head office router

1. **Create an ISAKMP policy.**

   The ISAKMP policy specifies the authentication method that identifies the remote router. It can be RSA encryption or a pre-shared key. This example uses a pre-shared key. First ensure you have defined a pre-shared key (see "Setting Security" on page 45-30).

   To create an ISAKMP policy that uses the shared key, use the commands:

   ```
   create isakmp policy=office_isakmp_policy peer=202.78.65.4
       authtype=preshared key=1 encalg=des hashalg=sha

   set isakmp policy=office_isakmp_policy senddeletes=on
       setcommitbit=on
   ```

2. **Create an SA specification.**

   ISAKMP can be configured to negotiate different algorithms by creating different SA specifications. For example, the IPsec ESP protocol supports various encryption algorithms. Depending on your configuration, ISAKMP makes a series of proposals to the peer about the encryption algorithms that ISAKMP, ESP, and AH use.

   In this example, the head office router is configured to choose 3DES2KEY in preference over DES. While AH supports MD5, DESMAC, and SHA as the hash algorithm, the router uses SHA.

   When creating IPsec SA specifications for ISAKMP key management, the **keymanagement** parameter must be set to ISAKMP. You need not specify SPI values and encryption key identities because ISAKMP/IKE negotiates them when creating actual SAs.

   To create an SA specification for the IPsec ESP protocol with DES encryption, use the command:

   ```
   create ipsec saspecification=1 keymanagement=isakmp
      protocol=esp encalg=des hashalg=null
   ```

   To create an SA specification for the IPsec ESP protocol with 3DES2KEY encryption, use the command:

   ```
   create ipsec saspecification=2 keymanagement=isakmp
      protocol=esp encalg=3des2key hashalg=null
   ```

   To create an SA specification for the IPsec AH protocol with the SHA hash algorithm, use the command:

   ```
   create ipsec saspecification=3 keymanagement=isakmp
      protocol=ah hashalg=sha
   ```

3. **Create a bundle specification.**

   Bundle specifications group together a set of SA specifications to create SA bundles. If you use a comma, you can specify several SA bundle proposals. Each bundle is proposed to the peer for consideration.

   To create a bundle specification, use the command:

   ```
   create ipsec bundlespecification=1 keymanagement=isakmp
      string="2 and 3,1 and 3"
   ```

4. **Create an IPsec policy.**

   Each IPsec policy has selector fields that define filtering rules for IP packets. IP packets are matched against these fields to determine the action to take. Possible actions are **ipsec**, **permit**, or **deny.** See "IPsec Policies" on page 45-14 for more information about actions.

   ISAKMP traffic must be permitted through the IPsec module so that the ISAKMP module can process it. To create an IPsec policy to permit ISAKMP traffic on the ppp0 interface, use the command:

   ```
   create ipsec policy=office_vpn_isakmp int=ppp0
      action=permit lport=500 rport=500
   ```

   By default ISAKMP traffic is received and transmitted over UDP port 500.

   To create a policy with an **ipsec** action, which encrypts your chosen payload traffic, use the command:

   ```
   create ipsec policy=office_vpn_ipsec int=ppp0 action=ipsec
      keymanagement=isakmp bundlespecification=1
      peer=202.78.65.4
   ```

Set the filter fields of the policy to protect data from the head office IP subnet 10.1.1.0 255.255.255.0 and destined for the remote site IP subnet 10.2.1.0 255.255.255.0 (Figure 45-3 on page 45-31):

```
set ipsec policy=office_vpn_ipsec laddress=10.1.1.0
    lmask=255.255.255.0 raddress=10.2.1.0
    rmask=255.255.255.0
```

5. Enable IPsec and ISAKMP processing.

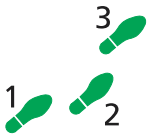
To enable IPsec and ISAKMP and activate the configuration, use the commands:

```
enable ipsec

enable isakmp
```

Reminder
It is important to create and set a boot-up configuration file that specifies a security officer. Use the commands:

```
create config=ipsec.cfg

set config=ipsec.cfg
```

Without this configuration file, no one can log in as security officer after a reboot. This means you would need to use the **disable system security** command to regain control of the router, but it automatically destroys keys.



To configure the remote router

1. Create an ISAKMP policy.

A matching ISAKMP policy must be created on the remote router for the head office router. First ensure you have defined a pre-shared key (see "Setting Security" on page 45-30).

To create an ISAKMP policy on the remote router, use the command:

```
create isakmp policy=office_isakmp_policy peer=202.45.12.8
    authtype=preshared key=1 encalg=des hashalg=sha

setisakmp policy=office_isakmp_policy senddeletes=on
    setcommitbit=on
```

2. Create an SA specification.

The SA specifications needed for the remote router are similar to the ones on the head office router. In this example, the remote router has no valid licence or resources to run 3DES encryption, so only DES encryption can be used. The identification numbers for the SA specifications can be different on both routers.

To create an SA specification for the IPsec ESP protocol, use the command:

```
create ipsec sas=1 keymanagement=isakmp protocol=esp
    keym=des hashalg=null
```

To create an SA specification for the IPsec AH protocol, use the command:

```
create ipsec sas=2 keymanagement=isakmp protocol=ah
    hashalg=sha
```

3. **Create bundle specifications.**

   To create a bundle specification on the remote router and group together the two SA specifications, use the command:

   ```
   create ipsec bundlespecification=1 keymanagement=isakmp
      sgring="1 and 2"
   ```

4. **Create an IPsec policy.**

   The remote router needs IPsec policies that match requirements of the head office IPsec policies.

   ISAKMP traffic must be permitted through the IPsec module so that the ISAKMP module can process it. To create an IPsec policy to permit ISAKMP traffic on the ppp0 interface, use the command:

   ```
   create ipsec policy=office_vpn_isakmp int=ppp0
      action=permit lport=500 rport=500
   ```

   By default ISAKMP traffic is received and transmitted over UDP port 500.

   To create a policy with an **ipsec** action, which encrypts your chosen payload traffic, use the command:

   ```
   create ipsec policy=office_vpn_ipsec int=ppp0 action=ipsec
      keymanagement=isakmp bundlespecification=1
      peer=202.45.12.8
   ```

   The **laddress** and **raddress** parameters determine the traffic to which the policy applies. They must be opposite to the ones set in the policy at the head office router:

   ```
   set ipsec policy=office_vpn_ipsec laddress=10.2.1.0
      lmask=255.255.255.0 raddress=10.1.1.0
      rmask=255.255.255.0
   ```

5. **Enable IPsec and ISAKMP processing.**

   To enable IPsec and ISAKMP and to activate the configuration, use the commands:

   ```
   enable ipsec

   enable isakmp
   ```

Reminder

It is important to create and set a boot-up configuration file that specifies a security officer. Use the commands:

```
create config=ipsec.cfg

set config=ipsec.cfg
```

Without this configuration file, no one can log in as security officer after a reboot. This means you would need to use the **disable system security** command to regain control of the router, but it automatically destroys keys.

VPN with NAT-Traversal

This example is a basic router-to-router solution with NAT-Traversal for a Virtual Private Network (VPN) that shows:

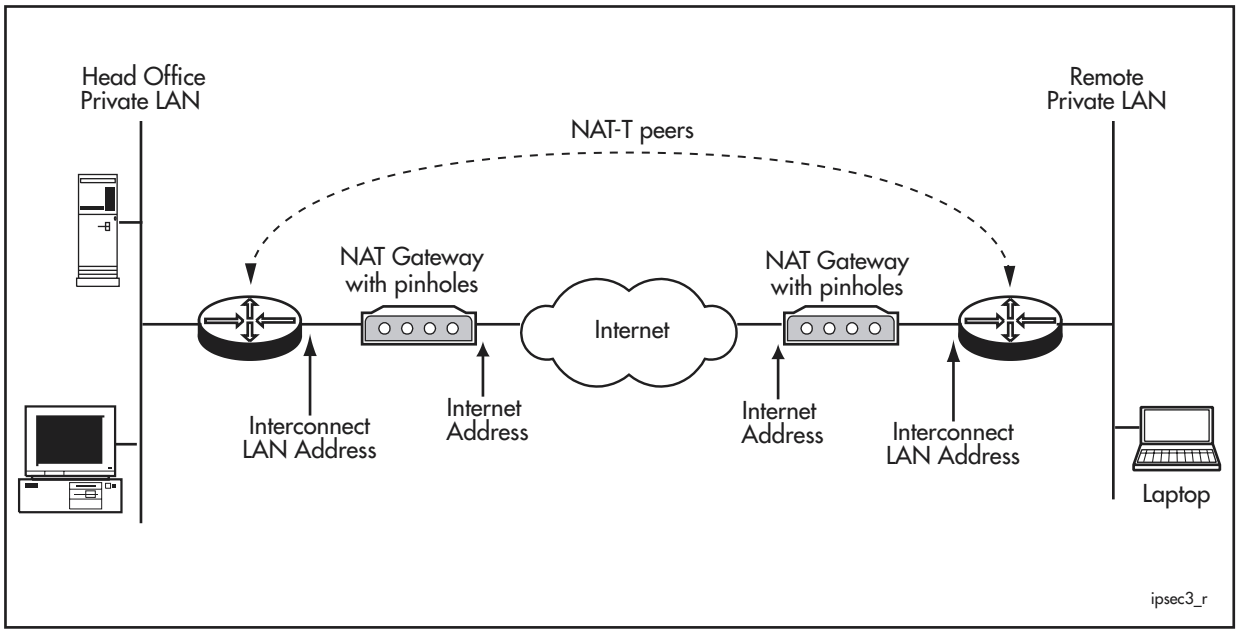
- NAT gateways at both ends of the VPN link
- a firewall configuration at both ends
- how to allow Internet access such as browsing

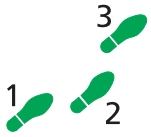
General Considerations

- This example also works with a NAT gateway at the initiator end, responder end, or neither end.
- IPsec and ISAKMP policies must refer to valid Internet peer addresses. When a peer router is directly connected to the Internet, this address is on its local WAN interface. When a peer accesses the Internet through a NAT gateway, this address is on the NAT gateway.
- If you have a NAT gateway at the responder end, it must be configured to allow traffic (*pinholes*) for UDP ports 500 and 4500.
- One end of the link must have a fixed Internet address. To enable both peers to initiate an IPsec link, both ends must have fixed addresses and both NAT gateways must have pinholes for UDP ports 500 and 4500.

  Many ISPs assign dynamic addresses that may change periodically, and you may need to ask for a fixed address.
- ISAKMP peers behind NAT gateways must identify themselves using a name string.
- If your office VPN router is behind an external gateway that does not use NAT, for example a firewall or IP filtering device, then the external gateway needs an ESP (protocol 50) permit rule in addition to UDP 500 and UDP 4500 permit rules. This allows NAT-T to work in most situations.
- If you configure your router remotely, we strongly recommend using Secure Shell. Telnet should not be used to access a secure gateway.

Figure 45-4: NAT-T router-to-router solution in IPsec tunnel mode





To configure the head office

1. **Set the router name.**

   First ensure you have defined a pre-shared key (see "Setting Security" on page 45-30), then use the commands:

   ```
   set system name="Head Office"
   ```

2. **Set security and authentication.**

   To add a security officer, use the commands:

   ```
   add user=secoff pass=secoff priv=sec login=yes telnet=yes

   delete user=manager
   ```

   During initial configuration when you are attending the router, we recommend that you set the security timeout period to 10 minutes. This avoids the inconvenience of logging in every time you pause. Use the command:

   ```
   set user securedelay=600
   ```

   After configuration, return the timeout to 60 seconds to maximise security. Use the command:

   ```
   set user securedelay=60
   ```

   ISAKMP Extended Authentication (XAUTH) is used in this example. To set the name and password by which to authenticate the IPsec peer, use the command:

   ```
   add user=remote password=friend
   ```

   Or for RADIUS server support for authentication, use the command:

   ```
   add radius server=radius-server-address secret=your-key
   ```

3. **Assign IP addresses to the interfaces.**

   Smaller MRU/MTU settings are needed for IPsec tunnel mode so that larger payload packets successfully pass through the IPsec tunnel.

   To enable IP, use the commands:

   ```
   set int=eth0 mtu=1300

   add ip int=eth0 ip=interconnect-LAN-address frag=yes

   add ip int=vlan1 ip=head-office-LAN-address

   add ip rou=0.0.0.0 mask=0.0.0.0 int=eth0
      next=NAT-gateway-address
   ```

4. **Enable a firewall.**

   To create a firewall policy and add to it, use the commands:

   ```
   create fire policy=main

   add fire policy=main int=vlan1 type=private

   add fire policy=main int=eth0 type=public

   add fire policy=main nat=enhanced int=vlan1 gblint=eth0
   ```

   To allow ISAKMP and NAT-T traffic on appropriate UDP ports, use the commands:

   ```
   add fire policy=main rule=1 int=eth0 action=allow
      ip=interconnect-LAN-address protocol=udp port=500
      gblip=interconnect-LAN-address gblpo=500
   ```

```
add fire policy=main rule=2 int=eth0 action=allow
    ip=interconnect-LAN-address protocol=udp port=4500
    gblip=interconnect-LAN-address gblpo=4500
```

To allow VPN traffic to bypass this firewall NAT, use the command:

```
add fire policy=main rule=3 int=eth0 action=nonat prot=all
    ip=head-office-LAN-ip-range x.x.x.x - x.x.x.x
    encap=ipsec
```

To add a NAT bypass rule for internally initiated VPN traffic to the remote site, use the command:

```
add firewall policy=main ru=4 action=nonat int=vlan1
    prot=all ip=head-office-LAN-ip-range x.x.x.x - x.x.x.x
    policy=main ru=4 remoteip=remote-LAN-ip-range
    x.x.x.x - x.x.x.x
```

5. Configure an SA specification.

The IPsec AH protocol is not used in this example but SHA is used as an alternative integrity check on the ESP protocol. DES encryption is used to encrypt the VPN payload.

To create an SA specification, use the commands:

```
create ipsec sas=1 key=isakmp protocol=esp encalg=des
    hasha=sha

create ipsec bundlespecification=1 key=isakmp string="1"
```

6. Create an IPsec policy.

To allow ISAKMP key negotiation and NAT-T processing to bypass IPsec, use the commands:

```
create ipsec policy=isakmp int=eth0 action=permit

set ipsec policy=isakmp lp=500

create ipsec policy=natt_udp int=eth0 action=permit

set ipsec policy=natt_udp lp=4500
```

To create a policy where the internet address of the peer is fixed, use the command:

```
create ipsec policy=remote_site int=eth0 action=ipsec
    key=isakmp bundlespecification=1
    peer=remote-internet-address isa=to_remote
```

Or when the remote router has a dynamically assigned IP address, use the command:

```
create ipsec policy=remote_site int=eth0 action=ipsec
    key=isakmp bundlespecification=1 peer=dynamic
    isa=to_remote
```

7. Set the IPsec policy and enable IPsec.

To set the IPsec policy to the remote router, use the command:

```
set ipsec policy=remote_site
    lad=head-office-LAN-ip-subnet-address
    lmask=head-office-LAN-ip-subnet-mask
    rad=remote-LAN-ip-subnet-address
    rmask=remote-LAN-ip-subnet-mask
```

For both VPN and internet-browsing access, use an **internet** policy. Do not use this for VPN-only access.

```
create ipsec policy=internet int=eth0 action=permit
```

To enable IPsec, use the command:

```
enable ipsec
```

8. **Create an ISAKMP policy.**

   To create a policy when the internet address of the peer is fixed, use the command:

   ```
   create isakmp policy=to_remote
      peer=remote-internet-address key=1
   ```

   Or when the peer has a dynamically assigned IP address, use the command:

   ```
   create isakmp policy=to_remote peer=any key=1
   ```

9. **Set the ISAKMP policy.**

   To set the ISAKMP policy to the remote router, use the command:

   ```
   set isakmp policy=to_remote localid=head_office
      heartbeat=receive
   ```

   ```
   set isakmp policy=to_remote sendd=true setc=true
   ```

   The heartbeat parameter is optional and lets the office delete inactive SAs if the Internet connection for the remote site drops. Heartbeats can be set to send, receive, or both. The receiver expects to receive heartbeats; when three are missing, it deletes the associated SA to avoid SA "out of step" fault conditions.

10. **Authenticate the IPsec peer.**

    To use Extended Authentication, use the command:

    ```
    set isakmp policy=to_remote xauth=server xauthtype=generic
    ```

11. **Enable ISAKMP key negotiation.**

    Use the command:

    ```
    enable isakmp
    ```

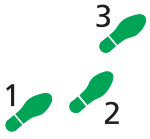
Reminder

It is important to create and set a boot-up configuration file that specifies a security officer. Use the commands:

```
create config=ipsec.cfg
```

```
set config=ipsec.cfg
```

Without this configuration file, no one can log in as security officer after a reboot. This means you would need to use the **disable system security** command to regain control of the router, but it automatically destroys keys.



To configure the remote router

1. **Set the router name.**

   First ensure you have defined a pre-shared key (see "Setting Security" on page 45-30), then use the commands:

   ```
   set system name="Remote Site"
   ```

2. **Set security and authentication.**

   To add a security officer, use the commands:

   ```
   add user=secoff pass=secoff priv=sec login=yes telnet=yes
   ```

```
delete user=manager
```

During initial configuration when you are attending the router, we recommend that you set the security timeout period to 10 minutes. This avoids the inconvenience of logging in every time you pause. Use the command:

```
set user securedelay=600
```

After configuration, return the timeout to 60 seconds to maximise security. Use the command:

```
set user securedelay=60
```

3. Assign IP addresses to the interfaces.

Smaller MRU/MTU settings are needed for IPsec tunnel mode so that larger payload packets successfully pass through the IPsec tunnel.

To enable IP, use the commands:

```
set int=eth0 mtu=1300

add ip int=eth0 ip=interconnect-LAN-address frag=yes

add ip int=vlan1 ip=remote-LAN-address

add ip rou=0.0.0.0 mask=0.0.0.0 int=eth0
   next=NAT-gateway-address
```

4. Enable a firewall.

To create a firewall policy and add rules to it, use the commands:

```
create fire poli=main

add fire policy=main int=vlan1 type=private

add fire policy=main int=eth0 type=public

add fire policy=main nat=enhanced int=vlan1 gblint=eth0
```

To allow ISAKMP and NAT-T traffic on appropriate UDP ports, use the commands:

```
add fire policy=main rule=1 int=eth0 action=allow
   ip=interconnect-LAN-address prot=udp port=500
   gblip=interconnect-LAN-address gblpo=500

add fire poli=main rule=2 int=eth0 action=allow
   ip=interconnect-LAN-address prot=udp port=4500
   gblip=interconnect-LAN-address gblpo=4500
```

To allow VPN traffic to bypass this firewall NAT, use the command:

```
add fire poli=main rule=3 int=eth0 action=nonat
   protocol=all ip=remote-LAN-ip-range x.x.x.x - x.x.x.x
   encap=ipsec
```

To add a rule for internally initiated VPN traffic to the head office, use the command:

```
add firewall policy=main rule=4 action=nonat int=vlan1
   protocol=all ip=remote-LAN-ip-range x.x.x.x - x.x.x.x
   policy=main rule=4 remoteip=head-office-LAN-ip-range
   x.x.x.x - x.x.x.x
```

5. Configure an SA specification.

To create an SA specification, use the commands:

```
create ipsec sas=1 key=isakmp protocol=esp encalg=des
   hasha=sha
```

```
create ipsec bundlespecification=1 key=isakmp string="1"
```

6. Create an IPsec policy.

To allow ISAKMP key negotiation and NAT-T processing to bypass IPsec, use the commands:

```
create ipsec policy=isakmp int=eth0 action=permit

set ipsec policy=isakmp lp=500 rp=500

create ipsec policy=natt_udp int=eth0 action=permit

set ipsec policy=natt_udp lp=4500 rp=4500
```

To create a policy when the internet address of the peer is fixed, use the command:

```
create ipsec policy=head_office int=eth0 action=ipsec
   key=isakmp bundlespecification=1
   peer=head-office-internet-address isa=to_office
```

Or when the peer has a dynamically assigned IP address, use the command:

```
create ipsec policy=head_office int=eth0 action=ipsec
   key=isakmp bundlespecification=1
   peer=dynamic isa=to_office
```

7. Set the IPsec policy and enable IPsec.

To set the IPsec policy to the head office, use the command:

```
set ipsec policy=head_office
   lad=remote-LAN-ip-subnet-address
   lmask=remote-LAN-ip-subnet-mask
   rad=head-office-LAN-ip-subnet-address
   rmask=head-office-LAN-ip-subnet-mask
```

For both VPN and internet-browsing access, use an **internet** policy. Do not use this for VPN-only access.

```
create ipsec policy=internet int=eth0 action=permit
```

To enable IPsec, use the command:

```
enable ipsec
```

8. Create an ISAKMP policy.

To create a policy where the internet address of the peer is on the peer's NAT gateway or its router, use the command:

```
create isakmp policy=to_office
   peer=head-office-internet-address key=1

set isakmp policy=to_office localid=remote_site
   heartbeat=send

set isakmp policy=to_office sendd=true setc=true
```

The heartbeat parameter is optional and lets the office delete inactive SAs if the Internet connection for the remote site drops. Heartbeats can be set to send, receive, or both. The receiver expects to receive heartbeats; when three are missing, it deletes the associated SA to avoid SA "out of step" fault conditions.

9. Authenticate the IPsec peer.

To use Extended Authentication, use the command:

```
set isakmp policy=to_office xauth=client xauthname=remote
   xauthpass=friend
```

10. Enable ISAKMP key negotiation.

Use the command:

```
enable isakmp
```

Reminder

It is important to create and set a boot-up configuration file that specifies a security officer. Use the commands:

```
create config=ipsec.cfg
set config=ipsec.cfg
```

Without this configuration file, no one can log in as security officer after a reboot. This means you would need to use the **disable system security** command to regain control of the router, but it automatically destroys keys.

Troubleshooting IPsec

Both IPsec and ISAKMP require the services of the ENCO module. Depending on the configuration of IPsec policies, IPsec may require the following resources:

- DES encryption
- 3DES encryption
- HMAC authentication
- STAC compression

ISAKMP may require:

- DES encryption
- 3DES encryption
- the Diffie-Hellman key exchange algorithm

To display the resources that the ENCO module can provide, use the command:

```
show enco
```

See Chapter 25, Compression and Encryption Services for information on configuring ENCO resources.

IPsec

If IPsec has not been enabled, it does not process IP packets. To display the status of IPsec, use the command:

```
show ipsec
```

IPsec events are logged in the router log. If IPsec is not working correctly, check the router log by using with the command:

```
show log
```

If the router log does not indicate why IPsec is failing to operate correctly, check the IPsec counters by using the command:

```
show ipsec counter
```

Before traffic can be processed by an IPsec policy, IPsec SAs and an SA bundle must have been created for that policy. To check whether an SA bundle has been created for a particular policy, use the command:

```
show ipsec policy=policy_name
```

To display the contents of the SA database and counters for an entry in the database, use the command:

```
show ipsed sa=sa-id [counter]
```

If an IPsec SA and bundle has not been created and key management is set to ISAKMP, then check the ISAKMP counters by using the command:

```
show isakmp counter
```

ISAKMP may not be able to negotiate an IPsec SA when IPsec configurations at each end of the link are incompatible.

If **keymanagement** is set to **manual** or **isakmp**, and counters suggest that ISAKMP was not able to negotiate an SA due to incompatible transforms, check the IPsec configuration at each end of the link by using the commands:

```
show ipsec saspecification

show ipsec bundlespecification

show ipsec policy
```

TO enable or disable IPsec debugging, use the commands:

```
enable ipsec policy[=name] debug={all|filter|packet}

disable ipsec policy={all|name} debug={all|trace}
```

Debugging for **filter** explains why packets are not being matched to a particular policy. Debugging for **trace** shows where a packet has failed in the IPsec process. Debugging for **packet** displays the contents of a packet at different stages of the IPsec process.

ISAKMP

If ISAKMP is not enabled, it does not listen on port 500 for ISAKMP messages. To display the status of ISAKMP, use the command:

```
show isakmp
```

ISAKMP events are logged in the router log. If ISAKMP is not working correctly, check the router log by using with the command:

```
show log
```

When the ISAKMP SA between two ISAKMP peers has been created, there are messages in the log indicating that a phase 1 ISAKMP exchange has started and completed successfully. The successful creation of an ISAKMP SA can be confirmed by viewing the SA in the ISAKMP SA database by using the command:

```
show isakmp sa
```

If the router log indicates that an ISAKMP phase 1 exchange has started but not finished, then the exchange may be waiting for the remote peer to reply. Until it times out and stops re-transmitting the last message, the exchange can still be displayed by using the command:

```
show isakmp exchange
```

If the router log does not indicate why an ISAKMP negotiation has failed, check the ISAKMP counters by using the command:

```
show isakmp counter
```

If the ISAKMP SA has not been created successfully, then check the counters for **main** mode. If the ISAKMP SA has been successfully created but IPsec SAs are not being negotiated, then check the counters for **quick** mode. In both cases the **general** counters may explain why ISAKMP messages are not being processed.

To enable or disable ISAKMP debugging, use the commands:

```
enable isakmp debug={all|default|packet|pkt|pktraw|state|
   trace|tracemore}

disable isakmp debug={all|default|packet|pkt|pktraw|state|
   trace|tracemore}
```

The **state** and **trace** options help determine the part of the ISAKMP negotiation that is failing. If these modes fail to reveal the problem, your support centre may ask that you capture text by using the **all** option.

The most common reason for the failure of a phase 1 ISAKMP exchange is that the pre-shared key or RSA public keys have not been configured correctly. Make sure that the pre-shared key is identical on each end of the link, and that the public key of each router is loaded onto the other router correctly.

The most common reason for the failure of a phase 2 ISAKMP exchange is that IPsec configurations at each end of the link are incompatible. Check that both of these are correct.

Command Reference

This section describes the commands available on the router to enable, configure, control, and monitor IPsec. IPsec requires IP routing and IP interfaces to be enabled and configured. See Chapter 14, Internet Protocol (IP) for the commands required to enable and configure IP routing and IP interfaces.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page xcv of Preface 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 error messages and their meanings.

Some interface and port types mentioned in this chapter may not be supported on your router. The interface and port types that are available vary depending on your product's model, and whether an expansion unit (PIC, NSM) is installed. For more information, see the Hardware Reference.

activate ipsec convertoldsa

Syntax

`ACTivate IPSec CONVERToldsa [SA=`*`sa-id`*`]`

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

Description

This command converts an old SA configuration in router memory to a functionality equivalent IPsec configuration in memory. IPsec must be enabled before this command can be used.

The **sa** parameter specifies the SA configuration to convert. If an SA is not specified, all old SAs are converted.

This command destroys the old SA configuration, and a warning message is displayed to let the user confirm the conversion.

When the conversion is complete, a message reminds the user to update the peer IP address of their policies and to save the configuration to a new boot configuration script.

Examples

To convert an old SA with the identification number 2 to the equivalent IPsec configuration, use the command:

```
act ips convert sa=2
```

To convert all old SA configurations to equivalent IPsec configurations, use the command:

```
act ips convert
```

Related Commands

disable ipsec oldsa
enable ipsec oldsa

add sa member

Syntax

```
ADD SA=sa-id MEMber={LOCal|REMote} IPaddress=ipadd
   MASK=ipadd
```

where:

- *sa-id* is a number from 0 to 255.
- *ipadd* is an IP address in dotted decimal notation.

Description

This command adds a member to a security association, and configures the router's pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration. This command is accepted only when the user has Security Officer privilege

The **member** parameter specifies whether the member is a local member or a remote member.

The **ipaddress** parameter specifies the base IP address of the member.

The **mask** parameter specifies the range of IP addresses that belong to the member.

Examples

To add a local member consisting of the IP addresses 192.168.1.1 to 192.168.1.15 to SA 1, use the command:

```
add sa=1 mem=local ip=192.168.1.1 mask=255.255.255.240
```

Related Commands

delete sa member
show sa

create ipsec bundlespecification

Syntax

```
CREate IPSec BUNDlespecification=bundlespecification-id
   KEYmanagement={ISakmp|MAnual} STRing="bundle-string"
   [EXPIRYKbytes=1..2000000000]
   [EXPIRYSeconds=300..31449600]
```

where:

- *bundlespecification-id* is a number from 0 to 255.
- *bundle-string* is a string 1 to 100 characters long, containing numbers from 0 to 255, and the separators *and*, *or*, and a comma.

Description

This command creates a bundle specification with the specified identification number in the IPsec Security Policy Database (SPD). Bundle specifications are templates to create SA bundles, and specify the number and order of SAs in a bundle. All SA bundles are created from a bundle specification.

The **bundlespecification** parameter specifies the identification number of the bundle. A bundle specification with the same identification number must not already exist.

The **keymanagement** parameter specifies whether the bundle specification is to be used manually to create an SA bundle or if it is to be used by ISAKMP/IKE to negotiate an SA bundle.

The **string** parameter lists SA specifications to use to create SAs in a bundle.

Bundle specifications used for manual key management can specify that the bundle consist of one, two, or three SAs. Each SA must use a different IPsec protocol (ESP, AH, or IPComp). The SA specification identification numbers in the bundle string that are used to create the SAs are separated by *ands*. The SAs are applied to outbound packets in the order of their respective SA specifications in the bundle string. They are applied in the reverse order on inbound packets.

Bundle specifications that ISAKMP/IKE uses can specify one or more "proposals" separated by commas. Each proposal can contain one, two, or three SAs. If different SA specifications offer a choice of algorithm(s) for the same IPsec protocol, they are separated by *or* in the bundle string. The order of SA specifications separated by *ors* determines the order of preference when used to negotiate an SA. SAs created from the negotiated proposal are applied to outbound packets in the order of their respective SA specifications in the bundle string. They are applied in the reverse order on inbound packets.

The **expirykbytes** parameter specifies the number of kilobytes of data that can be processed by the SAs in the bundle before the bundle expires and must be renegotiated. This parameter is valid when the **keymanagement** parameter is set to **isakmp**. By default, there is no limit on the number of kilobytes an SA can process in its lifetime.

The **expiryseconds** parameter specifies the maximum lifetime in seconds of the SAs in a bundle before the bundle expires and must be renegotiated. This parameter is valid when the **keymanagement** parameter is set to **isakmp**. The default is 28800 seconds (8 hours).

Examples

To create an SA bundle for use with manual key management, that creates two SAs based on SA specifications 1 and 2, use the command:

```
cre ips bund=1 key=ma str="1 and 2" expiryk=500000
```

The following command creates an SA bundle for use with ISAKMP key management. Two SA bundles are proposed; the first would create two SAs using either SA specification 1 or 2, and SA specification 3; and the second bundle would create two SAs based on SA specifications 4 and 3:

```
cre ips bund=2 key=is str="1 or 2 and 3, 4 and 3" expirys=7200
```

Related Commands

destroy ipsec bundlespecification
set ipsec bundlespecification
show ipsec bundlespecification

create ipsec policy

Syntax

```
CREate IPSec POLicy=name INTerface=interface ACtion={DEny|
   IPsec|PErmit} [IPVersion={4|6}]
   [BUNDlespecification=bundlespecification-id]
   [DFBit={SEt|COpy|CLear}] [GROup={0|1|2}]
   [ICmptype={list|NDALL}] [IPROUtetmplate=template-name]
   [ISAkmppolicy=isakmp-policy-name]
   [KEYmanagement={ISakmp|MAnual}] [LADdress={ANy|
   ipv4add[-ipv4add]|ipv6add[/prefix-length]|ipv6add-
   ipv6add}] [LMAsk=ipv4add] [LNAme={ANy|system-name}]
   [LPort={ANy|OPaque|port}] [PEERaddress={ipv4add|
   ipv6add|ANy|DYnamic}] [POSition=pos] [RADdress={ANY|
   ipv4add[-ipv4add]|ipv6add[/prefix-length]|ipv6add-
   ipv6add}] [RMAsk=ipv4add] [RNAme={ANy|system-name}]
   [RPort={ANy|port|OPaque}] [SASElectorfrompkt={ALL|
   LADdress|LPort|NONE|RADdress|RPort|TRAnsportprotocol}]
   [SRCInterface=interface] [TRAnsportprotocol={ANy|EGp|
   ESp|GRe|ICmp|OPaque|OSpf|RSvp|TCp|UDp|protocol}]
   [UDPHeartbeat={True|False}] [UDPPort=port]
   [UDPTunnel={True|False}] [USEPFSKey={True|False}]
```

where:

- *name* is a string 1 to 23 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.
- *interface* is an interface name formed by joining a layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15.
- *bundlespecification-id* is a number from 0 to 255.
- *list* is a comma-separated list of ICMP types.
- *template-name* is a string 1 to 31 characters long. It may contain any printable character and is case sensitive. If *template-name* contains spaces, it must be in double quotes.
- *isakmp-policy-name* is a string 1 to 24 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.
- *ipv4add* is an IPv4 address in dotted decimal notation.
- *ipv6add* is an IPv6 address in colon-separated hexadecimal notation, with its prefix length optionally indicated by slash notation.
- *prefix-length* is a number between 1 and 128.
- *system-name* is a string 1 to 119 characters long. Valid characters are any printable character. If *system-name* contains spaces, it must be in double quotes.
- *port* is an Internet service port number.
- *pos* is a number from 1 to 100.
- *protocol* is an Internet IP protocol number.

Description This command creates an IPsec policy with the specified name in the IPsec Security Policy Database (SPD).

The **policy** parameter specifies the name of the policy to create. A policy with the specified name must not already exist.

The **interface** parameter specifies the IP logical interface to which the policy is to be attached. The IP logical interface must exist. Valid interfaces are:

- eth (e.g. eth0, eth0-1)
- PPP (e.g. ppp0, ppp1-1)
- VLAN (e.g. vlan1, vlan0-1)
- FR (e.g. fr0, fr0-1)
- X.25 DTE (e.g. x25t0, x25t0-1)
- virtual tunnel (e.g. virt9)

To see a list of current valid interfaces, use the commands **show interface** command on page 7-66 of Chapter 7, Interfaces.

The **action** parameter specifies what action is to be performed on packets that match the policy. If **ipsec** is specified, matching packets are processed by an SA bundle attached to the policy. If **ipsec** is specified, the **peeraddress** parameter is required. If **permit** is specified, matching packets are allowed to bypass IPsec processing. If **deny** is specified, matching packets are discarded immediately.

The **bundlespecification** parameter specifies a bundle specification to be used when creating an SA bundle for this policy. The bundle specification must already exist. The key management specified in the bundle specification must match the one for the policy. This parameter is required when the **action** parameter for the policy is **ipsec**.

The **dfbit** parameter specifies the action taken on the DF bit in the outer IP header. This option is valid for tunnel mode operation. When this parameter is set to **copy**, the DF bit is copied from the inner IP header. When set to **set**, the DF bit in the outer IP header is set. When set to **clear**, the DF bit is cleared. If **ipversion** is 6, this parameter cannot be specified. The default is **clear**.

The **group** parameter specifies the group to be used for the Diffie-Hellman key exchange by ISAKMP/IKE for Perfect Forward Secrecy. Groups 1 and 2 are Oakley groups. This parameter can be used if the **usepfskey** parameter is set to **true**. The default is 1.

The **icmptype** parameter specifies the icmp type value of icmp packets to be matched against. The values can be specified as a comma-separated list, or by specifying **ndall**, which is equivalent to specifying types 133,134,135, and 136 (the types that are required for IPv6 neighbour discovery). This parameter can be used if **ipversion** is 6.

The **iproutetemplate** parameter specifies the name of an IP route template so that IPsec can add an IP route. This parameter is valid when the **peeraddress** is set to **any** or **dynamic,** which determines the actual IP address of a peer. If **ipversion** is 6, this parameter cannot be specified. The default is no template.

The **ipversion** parameter specifies the version of IP that all relevant addresses and network connections are to be checked against for validity. If **4** is specified, the version is IPv4. If **6** is specified, the version is IPv6. The default is 4.

The **isakmppolicy** parameter specifies the name of the ISAKMP policy to use to negotiate SA bundles with the IPsec peer. This parameter is required when the **action** parameter is set to **ipsec**.

The **keymanagement** parameter specifies the key management mechanism to be used when creating SA bundles for this policy. If **manual** is specified, SA bundles are created manually. If **isakmp** is specified, SA bundles are negotiated by ISAKMP/IKE with the IPsec peer. This parameter is required when the **action** parameter is **ipsec**.

The **laddress** and **lmask** parameters specify the selection value of the policy's local IP address selector. If **laddress** is set to **any**, the selection value is any IP address of the IP version specified by the **ipversion** parameter. If **laddress** specifies a single IPv4 address and **lmask** is not specified, the selection value is a single IPv4 address. If **laddress** specifies a single IPv4 address and **lmask** specifies a valid IP address mask, the selection value is a subnet of IPv4 addresses. If **ipversion** is 6, the **lmask** parameter cannot be specified. If **laddress** specifies a single IPv6 address and no prefix-length, the selection value is a single IPv6 address. If **laddress** specifies a single IP address with a valid prefix, the selection value is a subnet of IP addresses. If **laddress** specifies two IPv4 or IPv6 addresses separated by a "-", the selection value is a range of addresses. The default is **any**.

The **lname** parameter specifies the selection value of the policy's local name selector. The default is **any**.

The **lport** parameter specifies the selection value of the policy's local port selector. The default is **any**.

The **peeraddress** parameter specifies the IP address of the IPsec device acting as a peer for this IPsec policy. To allow multiple peers to connect simultaneously to this policy, set this parameter to **any**. For one peer with a dynamic address to connect to this policy, set the parameter to **dynamic**. This parameter is required when the **action** parameter is set to **ipsec.**

The **position** parameter specifies that the policy is to be attached to the IP logical interface in the specific position in the ordered list of policies for the interface. The default is to place the policy at the end of the list.

The **raddress** and **rmask** parameters specify the selection value of the policy's remote IP address selector. If **raddress** is set to **any**, the selection value is any IP address of the IP version specified by the **ipversion** parameter. If **raddress** specifies a single IPv4 address and **rmask** is not specified, the selection value is a single IPv4 address. If **raddress** specifies a single IPv4 address and **rmask** specifies a valid IP address mask, the selection value is a subnet of IPv4 addresses. If **ipversion** is 6, the **rmask** parameter cannot be specified. If **raddress** specifies a single IPv6 address and no prefix-length, the selection value is a single IPv6 address. If **raddress** specifies a single IP address with a valid prefix, the selection value is a subnet of IP addresses. If **raddress** specifies two IPv4 or IPv6 addresses separated by a "-", the selection value is a range of addresses. The default is **any**.

The **rname** parameter specifies the selection value of the policy's remote name selector. The default is **any**.

The **rport** parameter specifies the selection value of the policy's remote port selector. The default is **any**.

The **saselectorfrompkt** parameter specifies the selector value in the SAs used by this policy that is taken from processed packets rather than from the policy's value for the selector. More than one selector can be specified with a comma-separated list. The default is **none**.

The **srcinterface** parameter specifies which interface on the router to use as the source interface for tunnelled IPsec traffic. If **ipversion** is 6, this parameter cannot be specified. If the **srcinterface** parameter is not specified, the router defaults to the **interface** parameter.

The **transportprotocol** parameter specifies the selection value of the policy's transport protocol selector. The default is **any**.

The **udpheartbeat** parameter specifies whether UDP heartbeats should be sent. UDP heartbeats ensure that state information stored for the UDP tunnelled IPsec packets in intermediate devices is periodically refreshed. If **ipversion** is 6, this parameter cannot be specified. The default is **false**. If UDP tunnelling is not enabled, this parameter is ignored until UDP tunnelling is enabled.

The **udpport** parameter specifies the UDP port over which IPsec packets are to be tunnelled. If **ipversion** is 6, this parameter cannot be specified. The default port is 2746. If UDP tunnelling is not enabled, this parameter is ignored until UDP tunnelling is enabled.

The **udptunnel** parameter specifies that all traffic matching this policy should be tunnelled over UDP, and can be used when there is a gateway in the communication path that does not understand the data streams created by IPsec. If **ipversion** is 6, this parameter cannot be specified. The default is **false** because UDP tunnelling increases the packet overhead.

The **usepfskey** parameter specifies whether ISAKMP/IKE uses Perfect Forward Secrecy when creating keys for SAs used by the policy. This parameter is valid when the specified key management mechanism for the policy is **isakmp**. The default is **false**.

Examples

To create an IPsec policy, use the command:

```
cre ips pol="test" int=eth0 ac=ips peer=10.109.1.1 scri=eth1
   key=ma bund=1 rad=10.109.1.1
```

To create an IPsec policy with manual keying, use the command:

```
cre ips pol="my_vpn" int=ppp0 ac=ips peer=192.168.2.1 key=ma
   bund=1

set ips pol="my_vpn" lad=192.168.2.0 lma=255.255.255.0
   rad=192.167.2.0 rma=255.255.255.0
```

To create an IPsec policy with ISAKMP key management, use the command:

```
cre ips pol="my_vpn" int=ppp0 ac=ips key=isakmp bund=1
   peer=192.168.2.1

set ips pol="my_vpn" lad=192.168.2.0 lma=255.255.255.0
   rad=192.167.2.0 rma=255.255.255.0
```

Related Commands

destroy ipsec policy
disable ipsec policy debug
enable ipsec policy debug
set ipsec policy
show ipsec policy

create ipsec saspecification

Syntax

```
CREate IPSec SASpecification=spec-id
   KEYmanagement={ISAkmp|MAnual} PROTocol={AH|COmp|ESp}
   [ANTIReplayenabled={True|False}] [COMPalg=LZS]
   [ENCalg={3DES2key|3DESOuter|3DESInner|DES|AES128|
   AES192|AES256|NULl}] [ENCKey=key-id] [HASHAlg={DESMac|
   MD5|NULl|SHA}] [HASHKey=key-id] [INSPI=spi]
   [MODe={TRansport|TUnnel}] [OUTSPI=spi]
   [REPLAywindowsize={32|64|128|256}]
```

where:

- *spec-id* is a number from 0 to 255.
- *key-id* is a number from 0 to 65535.
- *spi* is a number from 256 to 4294967295.

Description

This command creates an SA specification to be used as a template when IPsec SAs are created by IPsec or ISAKMP. An SA specification must use **esp** (encryption), **ah** (authentication) or **comp** (compression) protocols. If manual key management is to be used, **inspi**, **outspi,** and **enckey** must be specified for an encryption SA; and **inspi**, **outspi** and **hashkey** must be specified for an authentication SA. If ISAKMP is to be used, ISAKMP must be enabled and configured. Only algorithms must be specified because ISAKMP negotiates suitable SPIs and keys.

The **saspecification** parameter specifies the identification number for the SA proposal. An SA specification with the specified identification number must not already exist.

The **keymanagement** parameter specifies whether the keys and SPIs are to be manually entered or negotiated by ISAKMP.

The **protocol** parameter specifies the IPsec protocol to be negotiated in this proposal.

The **antireplayenabled** parameter specifies whether the anti-replay mechanism is enabled for the specified SA. The default is **false**. This parameter is not valid for a compression SA or manual key management.

The **compalg** parameter specifies the compression algorithm to be negotiated in this proposal. This parameter is required if **protocol** is set to **comp**.

The **encalg** parameter specifies the encryption algorithm to be negotiated in this proposal. This parameter is required if **protocol** is set to **esp**.

The **enckey** parameter specifies the encryption key to be used by SAs created from this SA specification. The value identifies an existing key in the ENCO key database. This parameter is required if **protocol** is set to **esp** and **keymanagement** is set to **manual**.

The **hashalg** parameter specifies the hash algorithm to be negotiated in this proposal. This parameter is required if **protocol** is set to **ah**, or if ESP is to be used with authentication.

The **hashkey** parameter specifies the key to be used for authentication purposes by SAs created from this SA specification. The value identifies an

existing key in the ENCO key database. This parameter is required if **protocol** is set to **ah** or **esp**, and **keymanagement** is set to **manual**.

The **inspi** parameter specifies the Security Parameter Index to be used by SAs created from this SA specification for inbound traffic. This parameter is required if **protocol** is set to **ah** or **esp**, and **keymanagement** is set to **manual**.

The **mode** parameter specifies the mode of operation of the SA to be negotiated. The default is **tunnel**.

The **outspi** parameter specifies the Security Parameter Index to be used by SAs created from this SA specification for outbound traffic. This parameter is required if **protocol** is set to **ah** or **esp**, and **keymanagement** is set to **manual**.

The **replaywindowsize** parameter specifies the size of the anti-replay window. The default is 32 packets. This parameter is not valid for a compression SA or manual key management.

Examples To create an SA specification for manual key management for ESP using DES and MD5, use the command:

```
cre ips sas=1 prot=es key=ma inspi=300 outspi=400 enc=des
   hashalg=md5 enck=1 hashkey=101
```

To create an SA specification for manual key management for AH using MD5, use the command:

```
cre ips sas=2 prot=ah key=ma inspi=300 outspi=400 hasha=md5
   hashk=10
```

To create an SA specification for ISAKMP key management for ESP using 168-bit 3DES and MD5, use the command:

```
cree ips sas=1 prot=es key=is enc=3desi hasha=md5
```

To create an SA specification for ISAKMP key management for AH using MD5, use the command:

```
cre ips sas=5 prot=ah key=is hasha=md5
```

Related Commands

destroy ipsec saspecification
set ipsec saspecification
show ipsec saspecification

create isakmp policy

Syntax

```
CREate ISAkmp POLICY=name PEer={ipv4add|ipv6add|ANy}
   [AUTHType={PREshared|RSAEncr|RSASig}] [DELETedelay=10]
   [DHExponentlength=160..1023] [ENCalg={3DES2key|
   3DESInner|3DESOuter|DES|AES128|AES192|AES256}]
   [EXPIRYKbytes=1..1000] [EXPIRYSeconds=600..31449600]
   [GROup={0|1|2}] [HAShalg={SHa|MD5}]
   [HEARtbeatmode={Both|None|Receive|Send}]
   [HYBRIDxauth={ON|OFf|TRue|FAlse}] [IPVersion={4|6}]
   [KEY=0..65535] [LOCALID={ipv4add|ipv6add|domainname|
   user-domainname|dist-name}] [LOCALRsakey=0..65535]
   [MODe={MAIn|AGGressive}] [MSGRETrylimit=0..1024]
```

```
[MSGTImeout=1..86400]
[NATTraversal={ON|OFf|TRue|FAlse}]
[PHASE2xchglimit={NOne|1..1024}]
[POLICYFilename=filename] [PREnegotiate={ON|OFf|TRue|
FAlse}] [REMOTEId={ipv4add|ipv6add|domainname|
user-domainname|dist-name}] [SENDDeletes={ON|OFf|TRue|
FAlse}] [SENDNotify={ON|OFf|TRue|FAlse}]
[SENDIdalways={ON|OFf|TRue|FAlse}] [SETCommitbit={ON|
OFf|TRue|FAlse}] [SRCInterface=interface]
[XAUth={CLient|SErver|NONE}] [XAUTHName=username]
[XAUTHPasswd=password] [XAUTHType={GEneric|RAdius}]
```

where:

- *name* is a string up to 24 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.
- *ipv4add* is an IPv4 address in dotted decimal notation.
- *ipv6add* is an IPv6 address in colon-separated hexadecimal notation.
- *domainname* is a fully-qualified domain name in the format foo.bar.com.
- *user-domainname* is a user fully-qualified domain name in the format user@foo.bar.com.
- *dist-name* is an X.500 distinguished name, as described in **Distinguished Names (DN)** in Chapter 1, Operation.
- *filename* is a file name in the format `device:filename.type`. Invalid characters are * + = "| \ [ ] ; : ? / , < >, and wildcards are not allowed. Valid characters are:
  - uppercase and lowercase letters
  - digits (0–9)
  - the characters ~ ’ ! @ # $ % ^ & ( ) _ - { }

  The *device* variable is optional, and specifies the physical memory device on which the file is stored, which is flash. If *device* is specified, it must be separated from the rest of the file name by a colon (:). If *device* is not specified, the default is flash. The file extension *type* must be SCP or CFG.
- *interface* is an interface name formed by joining a layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15.
- *password* is the password to use for authentication 1 to 64 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.
- *username* is the username to use for authentication 1 to 64 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.

Description This command creates an ISAKMP policy. An ISAKMP policy specifies parameters for creating and responding to an ISAKMP exchange with a peer.

The **policy** parameter specifies the name of the policy to create. A policy with the specified name must not already exist.

The **peer** parameter specifies the IP address of the ISAKMP peer. If **any** is specified, then connections are accepted from any IP address of the IP version specified by the **ipversion** parameter.

The **authtype** parameter specifies the method to use to authenticate the ISAKMP peer. If **preshared** is specified, shared keys are used. If **rsaencr** is specified, RSA encryption is used. If **rsasig** is specified, RSA Signatures are used. The default is **preshared**.

The **deletedelay** parameter specifies the number of seconds between the completion of an ISAKMP exchange and the deletion of the ISAKMP exchange information. This prevents a deadlock if the last message that the device sends in the exchange is lost - the exchange will be complete on the device's side but not on the ISAKMP peer's side. If a retransmission is received from the ISAKMP peer during the **deletedelay** period, then the device will resend its last message again, allowing the exchange to complete on both sides. Normally the **setcommitbit** parameter can be used to prevent such a deadlock, but the **deletedelay** period can also protect against the final connected isakmp message being lost. The default is 10 seconds.

The **dhexponentlength** parameter specifies the length in bits of the Diffie-Hellman private exponent. A large private exponent increases the security of generated keys. A small private exponent shortens the time taken for the Diffie-Hellman key exchange. The minimum and default for all three Diffie-Hellman groups is 160 bits. The maximum allowable values are: 511 bits for group 0; 767 bits for group 1; and 1023 bits for group 2.

The **encalg** parameter specifies the ISAKMP encryption algorithm to be used to encrypt ISAKMP messages. The default is **des**.

The **expirykbytes** parameter specifies the number of kilobytes of data that can be processed by the ISAKMP SA created from this policy before the SA expires and must be re-negotiated. There is no default.

The **expiryseconds** parameter specifies the maximum lifetime in seconds of the ISAKMP SA created from this policy before the SA expires and must be re-created. The default is 86400 (24 hours).

The **group** parameter specifies the Diffie-Hellman group to use when negotiating session keys. The default is 1.

The **hashalg** parameter specifies the ISAKMP hash algorithm to use for authenticating ISAKMP messages. The default is **sha**.

The **heartbeatmode** parameter specifies the exchange of ISAKMP heartbeat messages. If **send** is specified, the router sends a message every 20 seconds. If **receive** is specified, the router checks heartbeat messages. If three messages in a row are not received, the receiving router deletes SAs belonging to the sending router. If **both** is specified, the router sends and receives heartbeat messages. If **none** is specified, heartbeat mode is not enabled. If the **ipversion** parameter is 6, the **heartbeatmode** parameter cannot be specified. The default is **none**.

The **hybridxauth** parameter specifies whether to use the hybrid form of extended authentication. This applies when the **authtype** parameter is set to **rsasig**. If **ipversion** is 6, the **hybridxauth** parameter cannot be specified. The default is **false**.

The **ipversion** parameter specifies the version of IP that all relevant addresses and network connections are to be checked against for validity. If **4** is specified, the version is IPv4. If **6** is specified, the version is IPv6. The default is 4.

The **key** parameter specifies the key identification number of the ENCO key to be used for authentication of the peer. When **authtype** is set to **preshared,** this parameter is required and specifies a **general** key shared by both ISAKMP peers. When **authtype** is set to **rsaencr**, this parameter specifies the RSA Public key of the ISAKMP peer; if no value is specified, a key matching the IP address of the peer is searched for in the ENCO key database.

The **localid** parameter specifies how the local device should identify itself to the remote peer. The user can specify the ID in the form of an IP address (e.g. 192.168.1.1, 3ffe::4), a fully-qualified domain name (e.g. foo.bar.com), a user fully-qualified domain name (e.g. user@foo.bar.com), or an X.500 distinguished name (e.g. "cn=user, o=mycompany, c=us"). The default is the local IP address. If the **authtype** parameter is set to **rsasig**, the router tries to use the system's distinguished name as the local ID. To set the system's distinguished name, use the **set system distinguishedname** command on page 1-123 of Chapter 1, Operation.

The **localsakey** parameter specifies the key identification number of the ENCO Private RSA key with which to authenticate the local router to the peer. This parameter is used for RSA encryption and RSA signature authentication. When **authtype** is set to **rsaencr** or **rsasig**, this parameter is required unless a default has been set with the **enable isakmp** command.

The **mode** parameter specifies **main** or **aggressive** mode for phase 1. The default is **main**.

The **msgretrylimit** parameter specifies the number of times an ISAKMP message is retransmitted. The default is 5.

The **msgtimeout** parameter specifies the number of seconds between the initial transmission of an ISAKMP message and the first retransmission. Subsequent retransmissions of the message occur at longer intervals. The default is 20 seconds.

The **nattraversal** parameter enables or disables NAT-T thereby letting peers negotiate a UDP-encapsulated mode so that IPsec traffic can flow through a NAT device. The default is **on**.

The **phase2xchglimit** parameter specifies the maximum number of phase 2 exchanges allowed over an ISAKMP SA created from this policy. The default is **none**.

The **policyfilename** parameter specifies the security policy to send to remote ISAKMP peers when requested. For this feature to work, the **policyserverenabled** parameter for the **enable isakmp** command must be set to **true**. This feature is designed for use with the AT-VPN Client for Windows. For more information on this parameter, refer to the AT-VPN Client documentation.

The **prenegotiate** parameter specifies whether to negotiate the ISAKMP SA at startup when the **enable isakmp** command is issued. The default is **false**. When **prenegotiate** is **true**, the **create isakmp policy** command must be entered before the **enable isakmp** command on page 45-69. In boot scripts the **create isakmp policy** command should appear before the **enable isakmp** command. When the **create isakmp policy** command appears after the **enable isakmp** command, prenegotiation does not work.

The **remoteid** parameter specifies how the remote device should be identified. The user can specify the ID in the form of an IP address (e.g. 192.168.1.1,

3ffe::4), a fully-qualified domain name (e.g. foo.bar.com), a user fully-qualified domain name (e.g. user@foo.bar.com), or an X.500 distinguished name (e.g. "cn=user, o=mycompany, c=us"). The default is the source IP address of ISAKMP messages from the peer.

The **senddeletes** parameter specifies whether to send Delete messages. Setting this parameter to **true** or **on** means that when an SA is deleted, ISAKMP notifies the ISAKMP peer that the SA is no longer valid. This ensures that traffic goes over valid SAs. Because some ISAKMP implementations do not support this parameter, the default is **false**.

The **sendidalways** parameter specifies whether ID messages are always sent when an ISAKMP SA is being negotiated. The default is **false**.

The **sendnotify** parameter specifies whether to send Notify Status and Error messages. The default is **false**.

The **setcommitbit** parameter specifies whether the commit bit is to be set when negotiating an ISAKMP SA. Setting this parameter to **true** or **on** ensures that traffic is not sent over an SA until confirmation of SA establishment is received from the ISAKMP peer. Because some ISAKMP implementations do not support the **setcommitbit** parameter, the default is **false**.

The **srcinterface** parameter specifies the local interface to which the policy is attached. If **ipversion** is 6, this parameter cannot be specified. Valid interfaces are:

- eth (e.g. eth0, eth0-1)
- PPP (e.g. ppp0, ppp1-1)
- VLAN (e.g. vlan1, vlan0-1)
- FR (e.g. fr0, fr0-1)
- X.25 DTE (e.g. x25t0, x25t0-1)

To see a list of current valid interfaces, use the commands **show interface** command on page 7-66 of Chapter 7, Interfaces.

The **xauth** parameter specifies whether extended authentication is to be used, and the role the router plays in it. If **server** is specified for this parameter, the router initiates the XAUTH exchange. If **client** is specified, the router expects an XAUTH request from the remote server. If **ipversion** is 6, this parameter cannot be specified. The default is **none**, specifying that extended authentication should not be used.

The **xauthname** parameter specifies the user name to be used for extended authentication when acting as the client. If **ipversion** is 6, this parameter cannot be specified.

The **xauthpasswd** parameter specifies the password to be used for extended authentication when acting as the client. If **ipversion** is 6, this parameter cannot be specified.

The **xauthtype** parameter specifies the type of authentication to use in the extended authentication exchange. If **ipversion** is 6, this parameter cannot be specified. The default is **generic**.

Examples To create an ISAKMP policy called "my_isakmp_policy" that negotiates with a peer with the IP address 192.168.2.1 and uses ENCO key 5 as a shared key for authentication, use the command:

```
cre isa pol="my_isakmp_policy" pe=192.168.2.1 autht=pre key=5
```

To create an ISAKMP policy called "my_isakmp_policy" that negotiates with a peer with the IP address 3ffe::4 and uses ENCO key 5 as a shared key for authentication, use the command:

```
cre isa pol="my_isakmp_policy" ipv=6 pe=3ffe::4 autht=pre
   key=5
```

Related Commands **destroy isakmp policy**
set isakmp policy
show isakmp policy

create sa

Syntax `CREate SA=`*`sa-id`* `ENCKey=`*`key-id`* `SPI=`*`spi`* `[ENCalg={DES|3DES2key|3DESInner}] [DIrection={IN|OUT|BOTH}]`

where:

- *sa-id* is a number from 0 to 255.
- *key-id* is a number from 0 to 65535.
- *spi* is a number from 256 to 4294967295.

Description This command creates a security association with the specified identification number, Security Parameter Index (SPI), and encryption key. This command configures pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration. Only users with Security Officer privilege can use this command.

The **sa** parameter specifies the identifier for the security association. A security association with the same identifier must not already exist on the router.

The **enckey** parameter specifies the identifier of the encryption key to be used by the security association. A key with the same identifier must exist.

The **spi** parameter specifies the Security Parameters Index for the security association. The SPI is used along with the destination address to identify a particular security association.

The **encalg** parameter specifies the DES encryption algorithm to be used by the security association. If **des** is specified, the standard DES encryption algorithm is used in CBC mode. If **3des2key** is specified, the Triple DES encryption algorithm is used in Outer CBC mode with two keys. If **3desinner** is specified, the Triple DES encryption algorithm is used in Inner CBC mode with three keys. The default is **des**.

The **direction** parameter specifies the direction of traffic passing through the IP interface to which the security association applies. If **in** is specified, incoming traffic is processed. If **out** is specified, outgoing traffic is processed. If **both** is

specified, the security association processes both incoming and outgoing traffic.

Examples To create an security association with an identification number of 1, with an SPI of 1000, encryption key 4, and that applies to all traffic on an interface, use the command:

```
cre sa=1 spi=1000 enck=4 di=both
```

Related Commands
destroy sa
set sa
show sa

delete sa member

Syntax `DELete SA=`*`sa-id`* `MEMber={LOCal|REMote} IPaddress=`*`ipadd`*
`MASK=`*`ipadd`*

where:

- *sa-id* is a number from 0 to 255.
- *ipadd* is an IP address in dotted decimal notation.

Description This command deletes an existing member of a security association. This command configures pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration. Only users with Security Officer privilege can use this command.

The **member** parameter specifies whether the member is local or remote.

The **ipaddress** parameter specifies the base IP address of the member.

The **mask** parameter specifies the range of IP addresses that belong to the member.

Examples To delete a local member consisting of the IP addresses 192.168.1.0 to 192.168.1.15 to SA 1, use the command:

```
del sa=1 mem=loc ip=192.168.1.1 mask=255.255.255.240
```

Related Commands
add sa member
show sa

destroy ipsec bundlespecification

Syntax `DESTroy IPSec BUNDlespecification=`*`bundlespecification-id`*

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

Description This command destroys the specified bundle specification.

The **bundlespecification** parameter specifies the identification number of the bundle. A bundle specification with the same identification number must already exist.

Examples To destroy bundle specification 1, use the command:

```
dest ips bund=1
```

Related Commands
create ipsec bundlespecification
set ipsec bundlespecification
show ipsec bundlespecification

destroy ipsec policy

Syntax `DESTroy IPSec POLicy=`*`name`* `[SABundle=`*`bundle-id`*`]`

where:

- *name* is a string 1 to 23 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.
- *bundle-id* is a number from 0 to 65535.

Description This command destroys an IPsec policy or one of its SA bundles. Destroying a policy also destroys the attached SA bundles, including active SA bundles.

The **policy** parameter specifies the name of the policy to destroy, or the name of the policy containing the bundle to destroy.

The **sabundle** parameter specifies the SA bundle to be destroyed. The SA bundle must exist in the specified policy.

Examples To destroy the IPsec policy named "my_vpn", use the command:

```
dest ips pol="my_vpn"
```

To destroy SA bundle 0 attached to the IPsec policy named "my_vpn", use the command:

```
dest ips pol= "my_vpn" sab=0
```

Related Commands
create ipsec policy
disable ipsec policy debug
enable ipsec policy debug

set ipsec policy
show ipsec policy

destroy ipsec saspecification

Syntax `DESTroy IPSec SASpecification=`*`spec-id`*

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

Description This command destroys a specific SA specification.

The **saspecification** parameter specifies the identification number of the specification to destroy. The specification must already exist.

Examples To destroy SA specification 1, use the command:

```
dest ips sas=1
```

Related Commands
create ipsec saspecification
set ipsec saspecification
show ipsec saspecification

destroy isakmp policy

Syntax `DESTroy ISAkmp POLicy=`*`name`*

where *name* is a string 1 to 24 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.

Description This command destroys the specified ISAKMP policy.

The **policy** parameter specifies the name of the policy to destroy. A policy with the same name must already exist.

Examples To destroy the ISAKMP policy named "my_isakmp_policy", use the command:

```
dest isa pol="my_isakmp_policy"
```

Related Commands
create isakmp policy
show isakmp policy

destroy sa

Syntax `DESTroy SA=`*`sa-id`*

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

Description This command destroys the security association with a specific identification number. This command configures pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration. Only users with Security Officer privilege can use this command.

The **sa** parameter specifies the identifier for the security association. An SA with the same identifier must exist on the router.

Examples To destroy security association 1, use the command:

```
dest sa=1
```

Related Commands
create sa
set sa
show sa

disable ipsec

Syntax `DISable IPSec`

Description This command disables IPsec processing on the router. All SA bundles attached to IPsec policies are deleted and IPsec detaches from the IP routing module. IPsec policies are not deleted.

Examples To disable IPsec processing on the router, use the command:

```
dis ips
```

Related Commands
disable ipsec policy debug
disable isakmp
enable ipsec
enable ipsec policy debug
enable isakmp
purge ipsec
show ipsec

disable ipsec oldsa

Syntax

```
DISable IPSec OLDsa INTerface=interface
```

where *interface* is an interface name formed by joining a layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15

Description

This command disables the use of the old SA functionality on the specified interface.

The **interface** parameter specifies the name of the interface. The interface must already be assigned to the IP routing module. Valid interfaces are:

- eth (e.g. eth0, eth0-1)
- PPP (e.g. ppp0, ppp1-1)
- VLAN (e.g. vlan1, vlan0-1)
- FR (e.g. fr0, fr0-1)
- X.25 DTE (e.g. x25t0, x25t0-1)

To see a list of current valid interfaces, use the commands **show interface** command on page 7-66 of Chapter 7, Interfaces.

Examples

To disable the old SA functionality on the ETH0 interface, use the command:

```
dis ips old int=eth0
```

Related Commands

activate ipsec convertoldsa
enable ipsec oldsa

disable ipsec policy debug

Syntax

```
DISable IPSec POLicy={ALl|name} DEBug={ALl|FIlter|PAcket|
   TRace} [DIrection={ALl|IN|OUt}] [DISPLAY={ALL|
   BUFFERHEADER|DATA|IPHEADER}] [RESult={ALL|FAIL|PASS}]
   [SELector={ALL|LADdress|LNAme|LPort|RADdress|RNAme|
   RPort|TRAnsportprotocol}] [WHEre={ALl|IPSec|PROTocol}]
```

where *name* is a string 1 to 23 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.

Description

This command disables debugging for the specified IPsec policy or all IPsec policies. For all parameters, multiple options may be specified as a comma-separated list.

The **policy** parameter specifies the name of the policy for which the debugging options are to be disabled. The specified policy must already exist.

The **debug** parameter specifies the debugging options to disable. If **filter** is specified, policy filter debugging is disabled. If **packet** is specified, packet debugging is disabled for all SAs used by the policy. If **trace** is specified, IPSEC trace debugging is disabled. If **all** is specified, all debugging options are disabled.

The **direction** parameter specifies the direction for which packet debugging is disabled. If **in** is specified, packet debugging is disabled for inbound traffic. If **out** is specified, packet debugging is disabled for outbound traffic. If **all** is specified, packet debugging is disabled for both inbound and outbound traffic. This parameter is not valid when filter debugging is specified.

The **display** parameter specifies the portions of packets not to be displayed. If **bufferheader** is specified, packet debugging does not display the original buffer header. If **ipheader** is specified, packet debugging does not display the IP header of the packet. If **data** is specified, packet debugging does not display the data portion of the packet. If **all** is specified, no packet debugging is displayed. This parameter is not valid when filter debugging is specified.

The **result** parameter specifies whether filter debugging is disabled for pass or fail results on selector fields. If **fail** is specified, filter debugging is disabled for packets with selector fields that do not match the policy. If **pass** is specified, the filter debugging is disabled for packets with selector fields matching the policy. If **all** is specified, all filter debugging is disabled. This parameter is not valid when filter debugging is specified.

The **selector** parameter specifies the packet selectors of the policy for which filter debugging is disabled. This parameter is not valid when filter debugging is specified.

The **where** parameter specifies the location or the processing unit where packet debugging is disabled. If **ipsec** is specified, packet debugging is disabled for the main IPsec processing unit. If **protocol** is specified, packet debugging is disabled for the relevant protocol processing unit. If **all** is specified, packet debugging is disabled for all processing units. This parameter is not valid when filter debugging is specified.

Examples

To disable policy filter debugging for the policy "my_vpn", use the command:

```
dis ips pol="my_vpn" deb=fi
```

Related Commands

enable ipsec policy debug
show ipsec policy

disable isakmp

Syntax

```
DISable ISAkmp
```

Description

This command disables ISAKMP processing on the router. All ISAKMP SAs and exchanges are destroyed. It has no effect on IPsec processing.

Examples

To disable ISAKMP processing on the router, use the command:

```
dis isa
```

Related Commands

disable isakmp debug
enable isakmp
enable isakmp debug
show isakmp

disable isakmp debug

Syntax

```
DISable ISAkmp DEBug={ALL|DEFault|PAcket|PKT|PKTRaw|STAte|
   TRAce|TRACEMore}
```

Description

This command disables the specified ISAKMP debugging options.

The **debugging** parameter specifies the debugging options to disable. The default is **default**, which disables trace, state, and packet debugging.

If **all** is specified, all debugging options are disabled. If **default** is specified, **trace**, **state**, and **packet** debugging are disabled.

If **state** is specified, debugging is disabled for changes in the ISAKMP state machine. If **packet** or **pkt** is specified, debugging is disabled for ISAKMP messages. If **pktraw** is specified, debugging is disabled for the raw hex contents of ISAKMP messages.

If **trace** is specified, debugging is disabled for ISAKMP error and informational messages. If **tracemore** is specified, debugging is disabled for ISAKMP calculated values.

Examples

To disable the default ISAKMP debugging modes (Trace, State and Packet), use the command:

```
dis isa deb
```

To disable ISAKMP trace debugging only, use the command:

```
dis isa deb=tra
```

Related Commands

disable isakmp
enable isakmp
enable isakmp debug
show isakmp

disable sa debug

Syntax `DISable SA=`*`sa-id`* `DEBug={All|Pkt|Search}`

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

Description This command disables the display of debugging information for the security association with the specified identification number. This command configures pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration. Only users with Security Officer privilege can use this command.

The **sa** parameter specifies the identifier for the security association. A security association with the specified identifier must exist on the router.

The **debug** parameter specifies the type of debugging information. If **all** is specified, all debugging is disabled. If **pkt** is specified, the display of packet debugging information is disabled. If **search** is specified, the display of IP address searching debugging information is disabled.

Examples To disable the display of all debugging information for security association 1, use the command:

```
dis sa=1 deb=a
```

Related Commands **enable sa debug**
set sa

enable ipsec

Syntax `ENAble IPSec`

Description This command enables IPsec processing on the router. If IPsec policies exist in the Security Policy Database (SPD), IPsec attaches to the IP module. SA bundles are created for policies with manual key management.

Examples To enable IPsec processing on the router, use the command:

```
ena ips
```

Related Commands **disable ipsec**
disable ipsec policy debug
disable isakmp
enable ipsec policy debug
enable isakmp
purge ipsec
show ipsec

enable ipsec oldsa

Syntax

```
ENAble IPSec OLDsa INTerface=interface
```

where *interface* is an interface name formed by joining a layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15

Description

This command enables the use of the old SA functionality on the given interface for backward compatibility.

The **interface** parameter specifies the name of the interface. The interface must already be assigned to the IP routing module. Valid interfaces are:

- eth (e.g. eth0, eth0-1)
- PPP (e.g. ppp0, ppp1-1)
- VLAN (e.g. vlan1, vlan0-1)
- FR (e.g. fr0, fr0-1)
- X.25 DTE (e.g. x25t0, x25t0-1)

To see a list of current valid interfaces, use the commands **show interface** command on page 7-66 of Chapter 7, Interfaces.

Examples

To enable the old SA functionality on the ETH0 interface, use the command:

```
ena ips old int=eth0
```

Related Commands

activate ipsec convertoldsa
disable ipsec oldsa

enable ipsec policy debug

Syntax

```
ENAble IPSec POLicy[=name] DEBug={ALl|FIlter|PAcket|TRace}
   [DIrection={ALl|IN|OUt}] [DISplay={ALl|BUFFERHEADER|
   DATA|IPHEADER}] [RESult={ALl|FAIl|PASS}]
   [SELector={ALl|LADdress|LNAme|LPort|RADdress|RNAme|
   RPort|TRAnsportprotocol}] [WHEre={ALl|IPSec|PROTocol}]
```

where *name* is a string 1 to 23 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.

Description

This command enables debugging for the specified IPsec policy or all IPsec policies. For all parameters, multiple options may be specified as a comma-separated list.

The **policy** parameter specifies the name of the policy for which the debugging options are to be enabled. The specified policy must already exist.

The **debug** parameter specifies the debugging options to enable. If **filter** is specified, policy filter debugging is enabled. If **packet** is specified, packet debugging is enabled for all SAs used by the policy. If **trace** is specified, IPSEC

trace debugging is enabled. If **all** is specified, all debugging options are enabled.

The **direction** parameter specifies the direction for which packet debugging is enabled. If **in** is specified, packet debugging is enabled for inbound traffic. If **out** is specified, packet debugging is enabled for outbound traffic. If **all** is specified, packet debugging is enabled for both inbound and outbound traffic. The default is **all**.

The **display** parameter specifies the portions of packets to be displayed. If **bufferheader** is specified, packet debugging displays the original buffer header. If **ipheader** is specified, packet debugging displays the IP header of the packet. If **data** is specified, packet debugging displays the data portion of the packet. If **all** is specified, packet debugging displays any of the original buffer header, IP header, or data portions of the packet. The default is **ipheader**, **data**.

The **result** parameter specifies whether filter debugging is enabled for pass or fail results on selector fields. If **fail** is specified, filter debugging is enabled for packets with selector fields that do not match the policy. If **pass** is specified, the filter debugging is enabled for packets with selector fields matching the policy. If **all** is specified, filter debugging is enabled for packets with both matching and unmatching selector fields. The default is **pass**.

The **selector** parameter specifies the packet selectors of the policy for which filter debugging is enabled.

The **where** parameter specifies the location or the processing unit where packet debugging is enabled. If **ipsec** is specified, packet debugging is enabled for the main IPsec processing unit. If **protocol** is specified, packet debugging is enabled for the relevant protocol processing unit. If **all** is specified, packet debugging is enabled for all processing units. The default is **ipsec**.

Examples

To enable policy filter debugging for the policy "my_vpn" use the command

```
ena ips pol="my_vpn" deb=fi
```

Related Commands

disable ipsec policy debug
show ipsec policy

enable isakmp

Syntax

```
ENAble ISAkmp [LOCALRsakey=key-id]
   [POLICYServerenabled={ON|OFf|TRue|FAlse}]
   [POLICYFilename=filename] [UDPPort=1..65535]
```

where:

- *key-id* is a number from 0 to 65535.
- *filename* is a file name in the format `device:filename.type`. Invalid characters are * + = " | \ [ ] ; : ? / , < >, and wildcards are not allowed. Valid characters are:
  - uppercase and lowercase letters
  - digits (0–9)
  - the characters ~ ' ! @ # $ % ^ & ( ) _ - { }

  The *device* variable is optional, and specifies the physical memory device on which the file is stored, which is flash. If *device* is specified, it must be separated from the rest of the file name by a colon (:). If *device* is not specified, the default is flash. The file *type* must be SCP or CFG.

Description

This command enables ISAKMP processing on the router. When ISAKMP policies exist with the **prenegotiate** parameter set to **true**, phase 1 exchanges start for those policies.

If the ISAKMP SA is to be prenegotiated at startup, then the **create isakmp policy** command on page 45-53 must be entered before **enable isakmp**. In boot scripts **create isakmp policy** should appear before the **enable isakmp** command. In both cases, the **enable isakmp** command must be the last ISAKMP command or prenegotiation does not work.

The **localrsakey** parameter specifies the ENCO key that is used as the router's RSA private key. This is required when RSAENCR authentication is used by ISAKMP policies.

The **policyserverenabled** parameter specifies whether the router acts as a security policy server. If it is set to **true** or **on**, the router listens for security policy requests from remote ISAKMP peers. The router responds by sending the security policy specified by the **policyfilename** parameter. The **policyfilename** parameter is required and an ISAKMP **policy** for the peer must exist. If **policyfilename** is specified on the matched ISAKMP policy, it is used. Otherwise, the **policyfilename** from the **enable isakmp** command is used. The default is **false**. This feature is designed for the AT-VPN Client for Windows. For more information on this parameter, refer to the AT-VPN Client product documentation.

The **policyfilename** parameter specifies the security policy sent to remote ISAKMP peers. This parameter specifies the default policy file and is used if the **policyfilename** parameter is not specified on the matched ISAKMP policy. This parameter must be specified when the **policyserverenabled** parameter is set to **true** or **on**. This feature is designed for the AT-VPN Client for Windows. For more information on this parameter, refer to the AT-VPN Client documentation.

The **udpport** parameter specifies the UDP port number where ISAKMP sends and receives messages. The default is 500.

Examples To enable ISAKMP processing on the router, use the command:

```
ena isa
```

Related Commands
disable isakmp
disable isakmp debug
enable isakmp debug
show isakmp

enable isakmp debug

Syntax

```
ENAble ISAkmp DEBug={ALl|DEFault|PAcket|PKT|PKTRaw|STAte|
   TRAce|TRACEMore}
```

Description This command enables the specified ISAKMP debugging options.

The **debugging** parameter specifies the debugging options to enable. The default is **default**, which enables trace, state, and packet debugging.

If **all** is specified, all debugging options are enabled. If **default** is specified, **trace**, **state**, and **packet** debugging are enabled.

If **packet** or **pkt** is specified, debugging is enabled for ISAKMP messages. If **pktraw** is specified, debugging is enabled for the raw hex contents of ISAKMP messages. If **state** is specified, debugging is enabled for changes in the ISAKMP state machine.

If **trace** is specified, debugging is enabled for ISAKMP error and informational messages. If **tracemore** is specified, debugging is enabled for ISAKMP calculated values.

Examples To enable default ISAKMP debugging modes (trace, state, packet), use the command:

```
ena isa deb
```

To enable ISAKMP trace debugging, use the command:

```
ena isa deb=tra
```

Related Commands
disable isakmp
disable isakmp debug
enable isakmp
show isakmp

enable sa debug

Syntax `ENAble SA=`*`sa-id`* `DEBug={All|Pkt|Search}`

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

Description This command enables the display of debugging information for the security association with the specified identification number. This command configures pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration. Only users with Security Officer privilege can use this command.

The **sa** parameter specifies the identifier for the security association. A security association with the specified identifier must exist on the router.

The **debug** parameter specifies the type of debugging information. If **all** is specified, all debugging is enabled. If **pkt** is specified, the display of packet debugging information is enabled. If **search** is specified, the display of IP address searching debugging information is enabled.

Examples To enable the display of all debugging information for security association 1, use the command:

```
ena sa=1 deb=a
```

Related Commands
disable sa debug
set sa

purge ipsec

Syntax `PURge IPSec`

Description This command destroys the existing IPsec configuration on the routers. All SA bundles, including active ones, are destroyed and active IPsec connections are closed.

Examples To destroy the IPsec configuration on the routers, use the command:

```
pur ips
```

Related Commands
destroy ipsec policy
disable ipsec
enable ipsec
show ipsec

reset ipsec counter

Syntax `RESET IPSec COUnter[={AH|ALG|COMp|ESp|MAIn|SAD|SETUP|SPD}]`

Description This command clears all general IPsec counters, or one or more categories of IPsec counters.

The **counter** parameter specifies the types of counters to clear. If a category is not specified, all general IPsec counters are cleared. Multiple categories can be specified as a comma-separated list.

If **ah** is specified, counters for the AH protocol are cleared.

If **alg** is specified, counters for the encryption and authentication algorithm processing units are cleared.

If **comp** is specified, counters for the IPComp protocol are cleared.

If **esp** is specified, counters for the IPsec ESP protocol are cleared.

If **main** is specified, counters for the main IPsec processing unit are cleared.

If **sad** is specified, counters for the IPsec Security Association Database are cleared.

If **setup** is specified, counters for setting up and removing IPsec SAs are cleared.

If **spd** is specified, counters for the IPsec security policy database are cleared.

Examples To clear the IPsec counters for SAD and SPD, use the command:

```
reset ips cou=sad,spd
```

To clear all IPsec counters, use the command:

```
reset ips cou
```

Related Commands **show ipsec counter**

reset ipsec policy counter

Syntax `RESET IPSec POLicy=`*`name`* `COUnter`

where *name* is a string 1 to 23 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.

Description This command clears counters for the specified IPsec policy. The **policy** parameter specifies the name of the policy to create. A policy with the same name must already exist.

Examples To clear the counters for the IPsec policy named "my_vpn_policy", use the command:

```
reset ips pol="my_vpn_policy" cou
```

Related Commands **show ipsec policy**

reset ipsec sa counter

Syntax `RESET IPSec SA=`*`sa-id`* `COUnter`

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

Description This command clears the counters of the specified Security Association (SA). The **sa** parameter specifies the identification number of the SA. The Security Association must already exist.

Examples To clear the counters for SA with identification number 1, use the command:

```
reset ips sa=1 cou
```

Related Commands **show ipsec sa**

reset isakmp counters

Syntax

```
RESET ISAkmp COUnters[={AGGressive|GENeral|HEArtbeat|INFo|
   IPSec|MAIn|NETwork|QUIck|SAD|SPD|TRAnsaction|XDB}]
```

Description This command clears all ISAKMP counters, or one or more categories of ISAKMP counters.

The **counters** parameter specifies the category or categories of counters to be cleared. If a counter category is not specified, all ISAKMP counters are cleared. Multiple categories can be specified as a comma-separated list.

If **aggressive** is specified, counters for Aggressive mode are cleared.

If **general** is specified, general ISAKMP counters are cleared.

If **heartbeat** is specified, counters for heartbeats are cleared.

If **info** is specified, counters for the informational exchanges are cleared.

If **ipsec** is specified, the interface counters between ISAKMP and IPSEC are cleared.

If **main** is specified, counters for the ISAKMP Main mode are cleared.

If **network** is specified, counters for the transmission of ISAKMP messages over the network are cleared.

If **quick** is specified, counters for the Quick mode are cleared.

If **sad** is specified, counters for the ISAKMP Security Association Database are cleared.

If **spd** is specified, counters for the ISAKMP Security Policy Database are cleared.

If **transaction** is specified, counters for transaction exchanges are cleared.

If **xdb** is specified, counters for the ISAKMP exchange database are cleared.

Examples To clear the **main** and **quick** ISAKMP counters, use the command:

```
reset isa cou=main,quick
```

To clear all ISAKMP counters, use the command:

```
reset isa cou
```

Related Commands **show isakmp counters**

set ipsec bundlespecification

Syntax

```
SET IPSec BUNDlespecification=bundlespecification-id
   [EXPIRYKbytes=1..2000000000]
   [EXPIRYSeconds=300..31449600]
```

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

Description This command modifies the bundle specification with the specified identification number in the IPsec Security Policy Database (SPD). Bundle specifications are a template for creating SA bundles, and specify the number and order of SAs that the bundle contains. All SA bundles are created from a bundle specification.

The **bundlespecification** parameter specifies the identification number of the bundle. A bundle specification with the same identification number must already exist.

The **expirykbytes** parameter specifies the number of kilobytes of data that can be processed by the SAs in the bundle before the bundle expires and must be renegotiated. This parameter can be specified when the **keymanagement** parameter is set to **isakmp**. By default there is no limit on the number of kilobytes an SA can process in its lifetime.

The **expiryseconds** parameter specifies the maximum lifetime in seconds of the SAs in the bundle before the bundle expires and must be renegotiated. This parameter can be specified when the **keymanagement** parameter is set to **isakmp**. The default is 28800 seconds (8 hours).

Examples To modify bundle specification 2 to expire after 2 hours, use the command:

```
set ips bund=2 expirys=7200
```

Related Commands

create ipsec bundlespecification
destroy ipsec bundlespecification
show ipsec bundlespecification

set ipsec policy

Syntax

```
SET IPSec POLIcy=name [ACtion={DEny|IPSec|PErmit}]
   [BUNDlespecification=bundlespecification-id]
   [DFBit={SEt|COpy|CLear}] [GROup={0|1|2}]
   [ICmptype={list|NDall}] [IPROUtetemplate=template-name]
   [IPVersion={4|6}] [ISAkmppolicy=isakmp-policy-name]
   [LADdress={ANy|ipv4add[-ipv4add]|ipv6add[/prefix-
   length]|ipv6add-ipv6add}] [LMAsk=ipv4add] [LNAme={ANy|
   system-name}] [LPort={ANy|OPaque|port}]
   [PEERaddress={ipv4add|ipv6add|ANy|DYNAMIC}]
   [POSition=pos] [RADdress={ANy|ipv4add[-ipv4add]|
   ipv6add[/prefix-length]|ipv6add-ipv6add}]
   [RMASK=ipv4add] [RNAme={ANy|system-name}] [RPort={ANy|
   port|OPaque}] [SRCInterface=interface]
   [TRAnsportprotocol={ANy|EGp|ESp|GRe|ICMP|OPaque|OSpf|
   RSvp|TCp|UDp|protocol}] [UDPHeartbeat={True|False}]
   [UDPPort=port] [UDPTunnel={True|False}]
   [USEPFSKey={True|False}]
```

where:

- *name* is a string 1 to 23 characters long. It may contain any printable character. If *name* contains spaces, it must be in double quotes.
- *bundlespecification-id* is a number from 0 to 255.
- *list* is a comma-separated list of icmp types.
- *template-name* is a string 1 to 31 characters long. It may contain any printable character and is case sensitive. If *template-name* contains spaces, it must be in double quotes.
- *isakmp-policy-name* is a string 1 to 24 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.
- *ipv4add* is an IPv4 address in dotted decimal notation.
- *ipv6add* is an IPv6 address in colon-separated hexadecimal notation, with its prefix length optionally indicated by slash notation.
- *prefix-length* is a number between 1 and 128.
- *system-name* is a string 1 to 119 characters long. Valid characters are any printable character. If *system-name* contains spaces, it must be in double quotes.
- *port* is an Internet service port number.
- *pos* is a number from 1 to 100.
- *interface* is an interface name formed by joining a layer 2 interface type, an interface instance, and optionally, a hyphen followed by a logical interface number from 0 to 15.
- *protocol* is an Internet IP protocol number.

Description This command modifies an IPsec policy with the specified name in the IPsec Security Policy Database (SPD).

The **policy** parameter specifies the name of the policy to create. A policy with the specified name must not already exist.

The **action** parameter specifies the action to perform on packets when they match the policy. If **ipsec** is specified, matching packets are processed by an SA bundle attached to the policy, and the **peeraddress** parameter is required. If **permit** is specified, matching packets are allowed to bypass IPsec processing. If **deny** is specified, matching packets are discarded immediately.

The **bundlespecification** parameter specifies a bundle specification to be used when creating an SA bundle for this policy. The bundle specification must already exist. The key management specified in the bundle specification must match the key management specified for the policy. This parameter is required when the **action** parameter is **ipsec**.

The **dfbit** parameter specifies the action taken on the DF bit in the outer IP header. This option is valid for tunnel mode operation. When this parameter is set to **copy**, the DF bit is copied from the inner IP header. When set to **set**, the DF bit in the outer IP header is set. When set to **clear**, the DF bit is cleared.

The **group** parameter specifies the group to be used for the Diffie-Hellman key exchange by ISAKMP/IKE for Perfect Forward Secrecy. Groups 1 and 2 are Oakley groups. This parameter can be used if the **usepfskey** parameter is set to **true**. The default is 1.

The **icmptype** parameter specifies the ICMP type value of ICMP packets to be matched against. The values can be specified as a comma-separated list, or by specifying **ndall**, which is equivalent to specifying types 133,134,135, and 136 (the types that are required for IPv6 neighbour discovery). This parameter can be used if **ipversion** is 6.

The **iproutetemplate** parameter specifies the name of an IP route template so that IPsec can add an IP route. This parameter is valid when the **peeraddress** is set to **any** or **dynamic,** which determines the actual IP address of a peer. If **ipversion** is 6, this parameter cannot be specified. The default is no template.

The **ipversion** parameter specifies the version of IP that all relevant addresses and network connections are to be checked against for validity. If **4** is specified, the version is IPv4. If **6** is specified, the version is IPv6. The default is 4.

The **isakmppolicy** parameter specifies the name of the ISAKMP policy to be used for negotiating SA bundles with the IPsec peer. This parameter is valid when the specified **action** parameter is **ipsec**.

The **laddress** and **lmask** parameters specify the selection value of the policy's local IP address selector. If **laddress** is set to **any**, the selection value is any IP address of the IP version specified by the **ipversion** parameter. If **laddress** specifies a single IPv4 address and **lmask** is not specified, the selection value is a single IPv4 address. If **laddress** specifies a single IPv4 address and **lmask** specifies a valid IP address mask, the selection value is a subnet of IPv4 addresses. If **ipversion** is 6, the **lmask** parameter cannot be specified. If **laddress** specifies a single IPv6 address and no prefix-length, the selection value is a single IPv6 address. If **laddress** specifies a single IP address with a valid prefix, the selection value is a subnet of IP addresses. If **laddress** specifies two IPv4 or IPv6 addresses separated by a "-", the selection value is a range of addresses. The default is **any**.

The **lname** parameter specifies the selection value of the policy's local name selector. The default is **any**.

The **lport** parameter specifies the selection value of the policy's local port selector. The default is **any**.

The **peeraddress** parameter specifies the IP address of the IPsec device acting as a peer for this IPsec policy. If the peer address is not known because it is dynamically assigned, set **peeraddress** to **dynamic**. If the policy can be used for any IPsec peer, set **peeraddress** to **any**. This parameter is required when the **action** parameter is **ipsec**.

The **position** parameter specifies that the policy is to be attached to the IP logical interface in the specific position in the ordered list of policies for the interface. The default is to place the policy at the end of the list.

The **raddress** and **rmask** parameters specify the selection value of the policy's remote IP address selector. If **raddress** is set to **any**, the selection value is any IP address. If **raddress** specifies a single IP address and **rmask** is not specified, the selection value is a single IP address. If **raddress** specifies a single IP address and **rmask** specifies a valid IP address mask, the selection value is a subnet of IP addresses. If **raddress** specifies two IP addresses separated by a "-", the selection value is a range of IP addresses. The default is **any**.

The **rname** parameter specifies the selection value of the policy's remote name selector. The default is **any**.

The **rport** parameter specifies the selection value of the policy's remote port selector. The default is **any**.

The **saselectorfrompkt** parameter specifies the selector value in the SAs used by this policy that is taken from processed packets rather than from the policy's value for the selector. More than one selector can be specified with a comma-separated list. The default is **none**.

The **srcinterface** parameter specifies which interface on the router is used as the source interface for tunnelled IPsec traffic. If **ipversion** is 6, this parameter cannot be specified.Valid interfaces are:

- eth (e.g. eth0, eth0-1)
- PPP (e.g. ppp0, ppp1-1)
- VLAN (e.g. vlan1, vlan0-1)
- FR (e.g. fr0, fr0-1)
- X.25 DTE (e.g. x25t0, x25t0-1)

If the **srcinterface** parameter is not specified, the router defaults to the interface that was specified using the **interface** parameter when the policy was created.

To see a list of current valid interfaces, use the commands **show interface** command on page 7-66 of Chapter 7, Interfaces.

The **transportprotocol** parameter specifies the selection value of the policy's transport protocol selector. The default is **any**.

The **udpheartbeat** parameter specifies whether UDP heartbeats should be sent. UDP heartbeats ensure that state information stored for the UDP tunnelled IPsec packets in intermediate devices is periodically refreshed. This parameter

cannot be specified for IPv6. The default is **false**. If UDP tunnelling is not enabled, this parameter is ignored until UDP tunnelling is enabled.

The **udpport** parameter specifies the UDP port over which IPsec packets are tunnelled. This parameter cannot be specified for IPv6. The default port is 2746. If UDP tunnelling is not enabled, this parameter is ignored until UDP tunnelling is enabled.

The **udptunnel** parameter specifies that all traffic matching this policy should be tunnelled over UDP, and can be used when there is a gateway in the communication path that does not understand the data streams created by IPsec. This parameter cannot be specified for IPv6. The default is **false** because UDP tunnelling increases the packet overhead.

The **usepfskey** parameter specifies whether ISAKMP/IKE uses Perfect Forward Secrecy when creating keys for SAs used by the policy. This parameter is valid when the specified key management mechanism for the policy is ISAKMP. The default is **false**.

Examples To set an IPsec policy, use the command:

```
set ips pol="test" int=eth0 ac=ips peer=10.109.1.1 srci=eth1
   key=ma bund=1 rad=10.109.1.1
```

To move the policy with the name "my_vpn" to position 3 on the interface, use the command:

```
set ips pol="my_vpn" pos=3
```

Related Commands
create ipsec policy
destroy ipsec policy
disable ipsec policy debug
enable ipsec policy debug
show ipsec policy

set ipsec saspecification

Syntax

```
SET IPSec SASpecification=spec-id
   [ANTIreplayenabled={True|False}] [COMPalg=LZS]
   [ENCalg={3DES2key|3DESInner|3DESOuter|DES|AES128|
   AES192|AES256|NULl}] [ENCKey=key-id] [HASHAlg={DESMac|
   MD5|NULl|SHA}] [HASHKey=key-id] [INSPI=spi]
   [MOde={TRansport|TUnnel}] [OUTSPI=spi]
   [REPLAywindowsize={32|64|128|256}]
```

where:

- *spec-id* is a number from 0 to 255.
- *key-id* is a number from 0 to 65535.
- *spi* is a number from 256 to 4294967295.

Description This command modifies an SA specification to be used as a template when IPsec SAs are created by IPsec or ISAKMP. An SA specification must use **esp** (encryption), **ah** (authentication) or **comp** (compression) protocols. If manual key management is to be used, **inspi**, **outspi,** and **enckey** must be specified for

an **esp** SA; and **inspi**, **outspi** and **hashkey** must be specified for an **ah** SA. If ISAKMP is to be used, ISAKMP must be enabled and configured. Only algorithms must be specified because ISAKMP negotiates suitable SPIs and keys.

The **saspecification** parameter specifies the identification number for the SA proposal. An SA specification with the specified identification number must not already exist.

The **antireplayenabled** parameter specifies whether the anti-replay mechanism is enabled for the specified SA. The default is **false**. This parameter is not valid for a compression SA or manual key management.

The **compalg** parameter specifies the compression algorithm to be negotiated in this proposal. This parameter is required if the **comp** protocol is specified.

The **encalg** parameter specifies the encryption algorithm to be negotiated in this proposal. This parameter is required if the **esp** protocol is specified.

The **enckey** parameter specifies the encryption key to be used by SAs created from this SA specification. The value identifies an existing key in the ENCO key database. This parameter is valid if **protocol** is set to **esp** and **keymanagement** is set to **manual**.

The **hashalg** parameter specifies the hash algorithm to be negotiated in this proposal. This parameter is valid if **protocol** is set to **ah**, or if ESP is to be used with authentication.

The **hashkey** parameter specifies the key to be used for authentication purposes by SAs created from this SA specification. The value identifies an existing key in the ENCO key database. This parameter is valid if **protocol** is set to **ah** or **esp,** and **keymanagement** is set to **manual**.

The **inspi** parameter specifies the Security Parameter Index to be used by SAs created from this SA specification for inbound traffic. This parameter is valid if **protocol** is set to **ah** or **esp**, and **keymanagement** is set to **manual**.

The **mode** parameter specifies the mode of operation of the SA to be negotiated. The default is **tunnel**.

The **outspi** parameter specifies the Security Parameter Index to be used by SAs created from this SA specification for outbound traffic. This parameter is valid if **protocol** is set to **ah** or **esp**, and **keymanagement** is set to **manual**.

The **replaywindowsize** parameter specifies the size of the anti-replay window. The default is 32 packets. This parameter is not valid for a COMP SA or manual key management.

Examples To change the algorithms and keys used by SA specification 1 with manual key management, use the command:

```
set ips sas=1 enc=3des2 hasha=sha enck=2
```

To change the encryption and hash algorithms used by SA specification 2 with ISAKMP key management, use the command:

```
set ips sas=2 enc=3desi hasha=sha
```

Related Commands
create ipsec saspecification
destroy ipsec saspecification
show ipsec saspecification

set ipsec udpport

Syntax

```
SET IPSec UDPPort=port
```

where *port* is an IP port number

Description This command changes the UDP port over which IPsec listens for UDP tunnelled IPsec traffic. This port is also the default port over which UDP encapsulated packets are sent to the remote peer. If this command is not used to set the listen port, then the routers listens on the default port of 2746.

The **udpport** parameter specifies the UDP listen port.

Example To set the UDP listen port to port 4000 (a non-default) use the command:

```
set ips udpp=4000
```

Related Commands
show ipsec

set isakmp policy

Syntax

```
SET ISAkmp POLicy=name [AUTHType={PREshared|RSAEncr|
   RSASig}] [DHExponentlength=160..1023] [DELETedelay=10]
   [ENCalg={3DES2key|3DESInner|3DESOuter|DES|AES128|
   AES192|AES256}] [EXPIRYKbytes=1..1000]
   [EXPIRYSeconds=600..31449600] [GROup={0|1|2}]
   [HAShalg={SHa|MD5}] [HEARtbeatmode={Both|None|Receive|
   Send}] [HYBRIDxauth={ON|OFf|TRue|FAlse}]
   [IPVversion={4|6}] [KEY=0..65535] [LOCALID={ipv4add|
   ipv6add|domainname|user-domainname|dist-name}]
   [LOCALRsakey=0..65535] [MODe={MAIn|AGGressive}]
   [MSGRetrylimit=0..1024] [MSGTImeout=1..86400]
   [NATTraversal={ON|OFf|TRue|FAlse}]
   [PHASE2xchglimit={NOne|1..1024}]
   [POLICYFilename=filename] [PREnegotiate={ON|OFf|TRue|
   FAlse}] [REMOTEId={ipv4add|ipv6add|domainname|user-
   domainname|dist-name}] [SENDDeletes={ON|OFf|TRue|
   FAlse}] [SENDNotify={ON|OFf|TRue|FAlse}]
   [SENDIdalways={ON|OFf|TRue|FAlse}] [SETCommitbit={ON|
   OFf|TRue|FAlse}] [SRCInterface=interface]
   [XAUTH={CLient|SErver|NOne}] [XAUTHName=username]
   [XAUTHPasswd=password] [XAUTHType={GEneric|RAdius}]
```

where:

- *name* is a string up to 24 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.
- *ipv4add* is an IPv4 address in dotted decimal notation.
- *ipv6add* is an IPv6 address in colon-separated hexadecimal notation.
- *domainname* is a fully-qualified domain name in the format foo.bar.com.
- *user-domainname* is a user fully-qualified domain name in the format user@foo.bar.com.
- *dist-name* is an X.500 distinguished name, as described in *"Distinguished Names (DN)" on page 1-50 of Chapter 1, Operation*.
- *filename* is a file name in the format `device:filename.type`. Invalid characters are * + = " | \ [ ] ; : ? / , < >, and wildcards are not allowed. Valid characters are:
  - uppercase and lowercase letters
  - digits (0–9)
  - the characters ~ ' ! @ # $ % ^ & ( ) _ - { }

  The *device* variable is optional and specifies the physical memory device on which the file is stored, flash. If *device* is specified, it must be separated from the rest of the file name by a colon (:). If *device* is not specified, the default is flash. The file *type* must be SCP or CFG.
- *interface* is an interface name formed by joining a layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15.
- *password* is the password to use for authentication 1 to 64 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.
- *username* is the username to use for authentication 1 to 64 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.

Description

This command modifies an existing ISAKMP policy. An ISAKMP policy specifies the parameters for creating and responding to an ISAKMP exchange with a peer.

The **policy** parameter specifies the name of the policy to modify. A policy with the specified name must already exist.

The **authtype** parameter specifies the method to authenticate the ISAKMP peer. If **preshared** is specified, shared keys are used. If **rsaencr** is specified, RSA encryption is used. If **rsasig** is specified, RSA signatures is used. The default is **preshared**.

The **dhexponentlength** parameter specifies the length in bits of the Diffie-Hellman private exponent. A large private exponent increases the security of generated keys. A small private exponent shortens the time taken for the Diffie-Hellman key exchange. The minimum and default for all three Diffie-Hellman groups is 160 bits. The maximum allowable values are 511 bits for group 0, 767 bits for group 1, and 1023 bits for group 2.

The **deletedelay** parameter specifies the number of seconds between the completion of an ISAKMP exchange and the deletion of the ISAKMP exchange information. This prevents a deadlock if the last message that the device sends

in the exchange is lost - the exchange will be complete on the device's side but not on the ISAKMP peer's side. If a retransmission is received from the ISAKMP peer during the **deletedelay** period, then the device will resend its last message again, allowing the exchange to complete on both sides. Normally the **setcommitbit** parameter can be used to prevent such a deadlock, but the **deletedelay** period can also protect against the final connected isakmp message being lost. The default is 10 seconds.

The **encalg** parameter specifies the ISAKMP encryption algorithm to encrypt ISAKMP messages. The default is **des**.

The **expirykbytes** parameter specifies the number of kilobytes of data that can be processed by the ISAKMP SA created from this policy before the SA expires and must be re-negotiated. There is no default.

The **expiryseconds** parameter specifies the maximum lifetime in seconds for the ISAKMP SA created from this policy before the SA expires and must be re-created. The default is 86400 (24 hours).

The **group** parameter specifies the group to use for the Diffie-Hellman key exchange by ISAKMP/IKE for Perfect Forward Secrecy. Groups 1 and 2 are Oakley groups. This parameter is valid when the **usepfskey** parameter is set to **true**. The default is 1.

The **hashalg** parameter specifies the ISAKMP hash algorithm to authenticate ISAKMP messages. The default is **sha**.

The **heartbeatmode** parameter specifies the exchange of ISAKMP heartbeat messages. If **send** is specified, the router sends a message every 20 seconds. If **receive** is specified, the router checks for heartbeat messages. If three messages in a row are not received, the receiving router deletes SAs belonging to the sending router. If **both** is specified, the router both sends and receives heartbeat messages. If **none** is specified, heartbeat mode is not enabled. This parameter cannot be specified for IPv6. The default is **none**.

The **hybridxauth** parameter specifies whether to use the hybrid form of extended authentication. This applies if the **authtype** parameter is set to **rsasig**. This parameter cannot be specified for IPv6. The default is **false**.

The **ipversion** parameter specifies the version of IP that all relevant addresses and network connections are to be checked against for validity. If **4** is specified, the version is IPv4. If **6** is specified, the version is IPv6. The default is 4.

The **key** parameter specifies the key identification number of the ENCO key to be used for authentication of the peer. When **authtype** is set to **preshared** this parameter is required and specifies a general key shared by both ISAKMP peers. When **authtype** is set to **rsasig**, this parameter is required and specifies the RSA Public key of the ISAKMP peer. When **authtype** is set to **rsaencr**, this parameter specifies the RSA Public key of the ISAKMP peer, or if no value is specified, then a key matching the IP address of the peer is searched for in the ENCO key database.

The **localid** parameter specifies how the local device should identify itself to the remote peer. The user can specify the ID in the form of an IP address (for example, 192.168.1.1, 3ffe::4), a fully-qualified domain name (for example, foo.bar.com), a user fully-qualified domain name (for example, user@foo.bar.com), or an X.500 distinguished name (for example, "cn=user, o=mycompany, c=us"). The default is the local IP address. If the **authtype** parameter is set to **rsasig**, the router tries to use the system's distinguished

name as the local ID. To set the system's distinguished name, use the **set system distinguishedname** command on page 1-123 of Chapter 1, Operation.

The **localrsakey** parameter specifies the key identification number of the ENCO Private RSA key to authenticate the local router to the peer. This parameter for **rsaencr** and **rsasig** authentication types. When **authtype** is set to **rsaencr** or **rsasig**, this parameter is required unless a default has been set with the **enable isakmp** command.

The **mode** parameter specifies whether the **main** or **aggressive** option is to be used for phase 1. The default is **main**.

The **msgretrylimit** parameter specifies the number of times an ISAKMP message is retransmitted. The default is 5.

The **msgtimeout** parameter specifies the number of seconds between the initial transmission of an ISAKMP message and the first retransmission. Subsequent retransmissions of the message occur at longer intervals. The default is 20 seconds.

The **nattraversal** parameter enables or disables NAT-T thereby letting peers negotiate a UDP-encapsulated mode so that IPsec traffic can flow through a NAT device. The default is **on**.

The **phase2xchglimit** parameter specifies the maximum number of phase2 exchanges allowed over an ISAKMP SA created from this policy. The default is **none**.

The **policyfilename** parameter specifies the security policy to be sent to remote ISAKMP peers when requested. For this feature to work, the **policyserverenabled** parameter for the **enable isakmp** command must be **true**. This feature is designed for use with the AT-VPN Client for Windows. For more information on this parameter, refer to the AT-VPN Client documentation.

The **prenegotiate** parameter specifies whether to negotiate ISAKMP SA at startup when the **enable isakmp** command is issued. The default is **false**.

If **prenegotiate** is **true**, the **set isakmp policy** command must be entered before the **enable isakmp** command on page 45-69. In boot scripts **set isakmp policy** should appear before the **enable isakmp** command. When **set isakmp policy** appears after the **enable isakmp** command, prenegotiation does not work.

The **remoteid** parameter specifies how the remote device should be identified. The user can specify the ID in the form of an IP address (e.g. 192.168.1.1, 3ffe::4), a fully-qualified domain name (e.g. foo.bar.com), a user fully-qualified domain name (e.g. user@foo.bar.com), or an X.500 distinguished name (e.g. "cn=user, o=mycompany, c=us"). The default is the source IP address of ISAKMP messages from the peer.

The **senddeletes** parameter specifies whether to send Delete messages. Setting this parameter to **true** or **on** means that when an SA is deleted, ISAKMP notifies the ISAKMP peer that the SA is no longer valid. This ensures that traffic goes over valid SAs. Because some ISAKMP implementations do not support this parameter, the default is **false**.

The **sendidalways** parameter specifies whether ID messages are always sent when an ISAKMP SA is being negotiated. The default is **false**.

The **sendnotify** parameter specifies whether to send Notify Status and Error messages. The default is **false**.

The **setcommitbit** parameter specifies whether the commit bit is to be set when negotiating an ISAKMP SA. Setting this parameter to **true** or **on** ensures that traffic is not sent over an SA until confirmation of SA establishment is received from the ISAKMP peer. Because some ISAKMP implementations do not support the **setcommitbit** parameter, the default is **false**.

The **srcinterface** parameter specifies the local interface to which the policy is attached. This parameter cannot be specified for IPv6. Valid interfaces are:

- eth (e.g. eth0, eth0-1)
- PPP (e.g. ppp0, ppp1-1)
- VLAN (e.g. vlan1, vlan0-1)
- FR (e.g. fr0, fr0-1)
- X.25 DTE (e.g. x25t0, x25t0-1)

To see a list of current valid interfaces, use the **show interface** command on page 7-66 of Chapter 7, Interfaces.

The **xauth** parameter specifies whether extended authentication is to be used. If **server** is specified, the router initiates the XAUTH exchange. If **client** is specified, the router expects an XAUTH request from the remote server. This parameter cannot be specified for IPv6. The default is **none**, specifying that extended authentication is not used.

The **xauthname** parameter specifies the username for extended authentication when acting as the client. This parameter cannot be specified for IPv6.

The **xauthpasswd** parameter specifies the password for extended authentication when acting as the client. This parameter cannot be specified for IPv6.

The **xauthtype** parameter specifies what type of authentication in the extended authentication exchange. This parameter cannot be specified for IPv6. The default is **generic**.

Examples

To modify an existing ISAKMP policy called "my_isakmp_policy" so that delete, notify status, and error messages are sent, use the command:

```
set isa pol="my_isakmp_policy" sendn=tr sendd=tr
```

Related Commands

create isakmp policy
destroy isakmp policy
show isakmp policy

set sa

Syntax

```
SET SA=sa-id [DEBUGPkttype={Short|Full}] [DIrection={IN|
   OUT|BOTH}] [ENCalg={DES|3DES2key|3DESInner}]
   [ENCKey=key-id] [SPI=spi]
```

where:

- *sa-id* is a number from 0 to 255.
- *key-id* is a number from 0 to 65535.
- *spi* is a number from 256 to 4294967295.

Description

This command sets the parameters of an existing security association with the specified identification number. This command configures pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration. Only users with Security Officer privilege can use this command.

The **sa** parameter specifies the identifier for the security association. A security association with the specified identifier must already exist on the router.

The **debugpkttype** parameter specifies the amount and type of debugging information to display for each packet when PKT debugging is enabled on a security association. If **short** is specified, 24 bytes of each packet are displayed. If **full** is specified, full debugging information about each packet is displayed.

The **direction** parameter specifies the direction of traffic passing through the IP interface to which the security association applies. If **in** is specified, incoming traffic is processed by the security association. If **out** is specified, outgoing traffic is processed by the security association. If **both** is specified, both incoming and outgoing traffic are processed by the security association.

The **encalg** parameter specifies the DES encryption algorithm to be used by the security association. If **des** is specified, the standard DES encryption algorithm is used in CBC mode. If **3des2key** is specified, the Triple DES encryption algorithm is used in Outer CBC mode with two keys. If **3desinner** is specified, the Triple DES encryption algorithm is used in Inner CBC mode with three keys.

The **enckey** parameter specifies the identifier of the encryption key to be used by the security association. A key with the specified identifier must exist.

The **spi** parameter specifies the Security Parameters Index for the security association. The SPI is used with the destination address to identify a particular security association.

Examples

To set the encryption key of SA 1 to key 5, use the command:

```
set sa=1 enckey=5
```

Related Commands

create sa
destroy sa
disable sa debug
enable sa debug
show sa

show ipsec

Syntax `SHow IPSec`

Description This command displays status information about IPsec (Figure 45-5, Table 45-1 on page 45-86).

Figure 45-5: Example output from the **show ipsec** command

```
IPSEC Module Configuration

  Module Status ......................... ENABLED
  IPsec over UDP
     Status ............................. OPEN
     Listen Port ........................ 2746
```

Table 45-1: Parameters in the output of the **show ipsec** command

Parameter	Meaning
Module Status	Whether the IPsec module is enabled or disabled.
IPsec over UDP	
Status	Whether the UDP tunnelling listen port is open or closed.
Listen Port	The UDP port over which UDP tunnelled IPsec packets can be received.

Examples To display the IPsec module status, use the command:

```
SH ips
```

Related Commands **disable ipsec**
enable ipsec

show ipsec bundlespecification

Syntax `SHow IPSec BUNDlespecitification[=`*`bundlespecification-id`*`]`

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

Description This command displays information about all bundle specifications or a specific one.

The **bundlespecification** parameter specifies the identification number of the bundle specification to be displayed. If a value is not specified summary information for all bundle specifications is displayed (Figure 45-6, Table 45-2 on page 45-87).

Figure 45-6: Example output from **show ipsec bundlespecification** command

```
Id KeyManagment ExpKBytes ExpSec  String
-------------------------------------------------------------
1  MANUAL       -         -       1 AND 3
2  ISAKMP       10000     86400   1 OR 2 AND 3
3  ISAKMP       100000    28800   1 AND 3, 2 AND 4
```

Table 45-2: Parameters in the output of the **show ipsec bundlespecification** command

Parameter	Meaning
Id	The number used to identify the bundle specification.
keymanagement	Whether the key management method is manual or ISAKMP.
ExpKBytes	The maximum number of kilobytes of data that SAs created by the bundle specification may process before they expire and must be re-negotiated.
ExpSec	The seconds in the SAs' lifetime created by the bundle specification before they expire and must be re-negotiated.
String	The list of SA specifications for this bundle as a string that contains *and*, *or*, spaces. commas, and integers.

If a value is specified, detailed information about the specified bundle specification is displayed (Figure 45-7 on page 45-87, Table 45-3 on page 45-88).

Figure 45-7: Example output from the **show ipsec bundlespecification** command for a specific bundle.

```
Bundle Specification

Id ................... 2
Key Management ....... ISAKMP
String ............... 1 OR 2 AND 3
Expiry Kbytes ........ -
Expiry Seconds ....... 7200

Number of Proposals.............. 2

  Proposal ....................... 1
      Proposal Number ............ 1
      Protocol ................... ESP
      Number Of Transfroms ....... 2
               Transform 0 ........... SA Spec Id    1
               Transform 1 ........... SA Spec Id    2


  Proposal ....................... 2
      Proposal Number ............ 1
      Protocol ................... AH
      Number Of Transfroms ....... 1
               Transform 0 ........... SA Spec Id    3
```

Table 45-3: Parameters in the output of the **show ipsec bundlespecification** command for a specific bundle.

Parameter	Meaning
Id	The number used to identify this bundle specification.
Key Management	Whether the key management method is manual or ISAKMP.
String	The list of SA specifications for this bundle, as a string that contains *and*, *or*, spaces. commas, and integers.
ExpiryKBytes	The maximum number of kilobytes of data that SAs created by the bundle specification may process before they expire and must be re-negotiated.
ExpirySeconds	The lifetime, in seconds, of SAs created by the bundle specification before they expire and must be re-negotiated.
Num of Proposals	The number of proposals in this bundle specification.
Proposal	The identification number for a proposal.
Proposal Number	The proposal number. The same proposal number may appear in multiple proposals and implies a logical AND operation (e.g. protocol 1 AND protocol 2).
Protocol	Whether the protocol to use for this proposal is ESP, AH, or COMP.
Number of Transforms	The number of transforms in this proposal.
Transform	The transform number and the SA specification to use for this transform.

Examples To display all bundle specifications, use the command:

```
show ipsec bundlespecification
```

To display bundle specification 1, use the command:

```
show ipsec bundlespecification=1
```

Related Commands

create ipsec bundlespecification
destroy ipsec bundlespecification
set ipsec bundlespecification

show ipsec counter

Syntax

```
SHow IPSec COUnter[={AH|ALG|COMp|ESp|MAIn|SAD|SETUP|SPD}]
```

Description This command displays all information counters for IPsec, or one or more categories of IPsec counters.

The **counter** parameter specifies the category or categories of counters to be displayed. If a counter category is not specified, all general IPsec counters are displayed. Multiple categories can be specified as a comma-separated list.

If **ah** is specified, counters for the IPsec AH protocol are displayed (Figure 45-8 on page 45-89, Table 45-4 on page 45-89).

If **alg** is specified, counters for the encryption and authentication algorithm processing units are displayed (Figure 45-9 on page 45-91, Table 45-5 on page 45-92).

If **comp** is specified, counters for the IPComp protocol are displayed (Figure 45-10 on page 45-97, Table 45-6 on page 45-97).

If **esp** is specified, counters for the IPsec ESP protocol are displayed (Figure 45-11 on page 45-99, Table 45-7 on page 45-99).

If **main** is specified, counters for the main IPsec processing unit are displayed (Figure 45-12 on page 45-101, Table 45-8 on page 45-101).

If **sad** is specified, counters for the IPsec Security Association Database are displayed (Figure 45-13 on page 45-102, Table 45-9 on page 45-102).

If **setup** is specified, counters for setting up and removing IPsec SAs are displayed (Figure 45-14 on page 45-102, Table 45-10 on page 45-103).

If **spd** is specified, counters for the IPsec security policy database are displayed (Figure 45-15 on page 45-103, Table 45-11 on page 45-104).

Figure 45-8: Example output from the **show ipsec counter=ah** command

```
AH setup/remove counters
  setupGetSaFailed           0    setupFailed              0
  setupDone                  0
  removeGetSaFailed          0    removeNothingDone        0
  removeDone                 0

AH outbound processing counters
  bufChainCopy               0    seqNumberCycled          0
  hashStart                  0    hashFailImm              0
  hashDoneGetSaFail          0    hashFail                 0
  hashDoneSaBadState         0    hashGood                 0

AH inbound processing counters
  bufChainCopy               0    replayedPacket           0
  icvInvalid                 0    badPayloadLength         0
  hashStart                  0    hashFailImm              0
  hashDoneGetSaFail          0    hashFail                 0
  hashDoneSaBadState         0    hashGood                 0
```

Table 45-4: Parameters in the output of the **show ipsec counter=ah** command

Parameter	Meaning
AH setup/remove counter	Counter for setting up and removing AH SAs.
setupGetSaFailed	The number of times an AH SA setup was aborted because the SA no longer existed.
setupDone	The number of times an AH SA setup was completed successfully.
removeGetSaFailed	The number of attempts to remove an AH SA was aborted because the SA no longer existed.
removeDone	The number of times an AH SA was removed successfully.
setupFailed	The number of times an AH SA setup failed.

Table 45-4: Parameters in the output of the **show ipsec counter=ah** command

Parameter	Meaning
removeNothingDone	The number of times the removal of an AH SA required no action.
AH outbound processing counter	Counter for the processing of outbound AH SAs.
bufChainCopy	The number of outbound packets copied from a chain to a buffer prior to AH processing.
hashStart	The number of times AH hash processing started on an outbound packet.
hashDoneGetSaFail	The number of times the generation of an AH hash was aborted because the SA no longer existed.
hashDoneSaBadState	The number of times the generation of an AH hash was aborted because the SA was no longer valid.
seqNumberCycled	The number of times an outbound packet caused an AH sequence number to cycle.
hashFailImm	The number of times the generation of an AH hash failed immediately.
hashFail	The number of times the generation of an AH hash failed.
hashGood	The number of AH hashes generated successfully.
AH inbound processing counter	Counter for the processing of inbound AH SAs.
bufChainCopy	The number of inbound packets copied from a chain to a buffer prior to AH processing.
icvInvalid	The number of inbound AH packets seen with an invalid ICV.
hashStart	The number of times AH hash processing started on an inbound packet.
hashDoneGetSaFail	The number of times the generation of an AH hash was aborted because the SA no longer existed.
hashDoneSaBadState	The number of times the generation of an AH hash was aborted because the SA was no longer valid.
replayedPacket	The number of inbound ESP packets seen with an invalid sequence number.
badPayloadLength	The number of inbound ESP packets seen with an invalid payload length.
hashFailImm	The number of times the generation of an AH hash failed immediately.
hashFail	The number of times the generation of an AH hash failed.
hashGood	The number of AH hashes generated successfully.

Figure 45-9: Example output from the **show ipsec counter=alg** command

```
General Algorithm Counters
  nullKeymatProcessed       10
DES:
  desKeymatProcessed        10
  desAttachFail              0    desAttachGood             10
  desConfigureFail           0    desConfigureGood          10
  desRemove                  5    desDetached                5
  desEncodeGood             34    desDecodeGood             26
  desEncodeFail              0    desDecodeFail              0
  desEncodeDiscard           0    desDecodeDiscard           0
  desEncodeGetInfoFail      0
TRIPLE DES INNER:
 3DesInnerKeymatProcessed  0
 3DesInnerAttachFail        0   3DesInnerAttachGood        0
 3DesInnerConfigureFail     0   3DesInnerConfigureGood     0
 3DesInnerRemove            0   3DesInnerDetached          0
 3DesInnerEncodeGood        0   3DesInnerDecodeGood        0
 3DesInnerEncodeFail        0   3DesInnerDecodeFail        0
 3DesInnerEncodeDiscard     0   3DesInnerDecodeDiscard     0
 3DesInnerEncGetInfoFail    0
TRIPLE DES OUTER:
 3DesOuterKeymatProcessed  0
 3DesOuterAttachFail        0   3DesOuterAttachGood        0
 3DesOuterConfigureFail     0   3DesOuterConfigureGood     0
 3DesOuterRemove            0   3DesOuterDetached          0
 3DesOuterEncodeGood        0   3DesOuterDecodeGood        0
 3DesOuterEncodeFail        0   3DesOuterDecodeFail        0
 3DesOuterEncodeDiscard     0   3DesOuterDecodeDiscard     0
 3DesOuterEncGetInfoFail    0
TRIPLE DES 2KEY:
 3Des2KeyKeymatProcessed    0
 3Des2KeyAttachFail         0   3Des2KeyAttachGood         0
 3Des2KeyConfigureFail      0   3Des2KeyConfigureGood      0
 3Des2KeyRemove             0   3Des2KeyDetached           0
 3Des2KeyEncodeGood         0   3Des2KeyDecodeGood         0
 3Des2KeyEncodeFail         0   3Des2KeyDecodeFail         0
 3Des2KeyEncodeDiscard      0   3Des2KeyDecodeDiscard      0
 3Des2KeyEncGetInfoFail     0
AES:
 aesKeymatProcessed         0
 aesAttachFail              0    aesAttachGood              0
 aesConfigureFail           0    aesConfigureGood           0
 aesRemove                  0    aesDetached                0
 aesEncodeGood              0    aesDecodeGood              0
 aesEncodeFail              0    aesDecodeFail              0
 aesEncodeDiscard           0    aesDecodeDiscard           0
 aesEncGetInfoFail          0
SHA:
 shaKeymatProcessed         0
 shaAttachFail              0    shaAttachGood              0
 shaConfigureFail           0    shaConfigureGood           0
 shaRemove                  0    shaDetached                0
 shaEncodeGood              0    shaDecodeGood              0
 shaEncodeFail              0    shaDecodeFail              0
 shaEncodeDiscard           0    shaDecodeDiscard           0
```

Figure 45-9: Example output from the **show ipsec counter=alg** command (continued)

```
MD5:
 md5KeymatProcessed         0
 md5AttachFail              0    md5AttachGood              0
 md5ConfigureFail           0    md5ConfigureGood           0
 md5Remove                  0    md5Detached                0
 md5EncodeGood              0    md5DecodeGood              0
 md5EncodeFail              0    md5DecodeFail              0
 md5EncodeDiscard           0    md5DecodeDiscard           0
DES-MAC:
 desmacKeymatProcessed      0
 desmacAttachFail           0    desmacAttachGood           0
 desmacConfigureFail        0    desmacConfigureGood        0
 desmacRemove               0    desmacDetached             0
 desmacEncodeGood           0    desmacDecodeGood           0
 desmacEncodeFail           0    desmacDecodeFail           0
 desmacEncodeDiscard        0    desmacDecodeDiscard        0
LZS:
 lzsKeymatProcessed         0
 lzsAttachFail              0    lzsAttachGood              0
 lzsConfigureFail           0    lzsConfigureGood           0
 lzsRemove                  0    lzsDetached                0
 lzsEncodeGood              0    lzsDecodeGood              0
 lzsEncodeFail              0    lzsDecodeFail              0
 lzsEncodeDiscard           0    lzsDecodeDiscard           0
IPSec:
ipsecKeymatProcessed        0
 ipsecAttachFail            0    ipsecAttachGood            0
 ipsecConfigureFail         0    ipsecConfigureGood         0
 ipsecRemove                0    ipsecDetached              0
 ipsecEncodeGood            0    ipsecDecodeGood            0
 ipsecEncodeFail            0    ipsecDecodeFail            0
 ipsecEncodeDiscard         0    ipsecDecodeDiscard         0
```

Table 45-5: Parameters in the output of the **show ipsec counter=alg** command

Parameter	Meaning
nullKeymatProcessed	The number of times NULL key material has been processed.
DES	Counter for the DES algorithm processing unit.
desKeymatProcessed	The number of times DES key material has been processed.
desAttachFail	The number of times a DES attach failed.
desConfigureFail	The number of times a DES configure failed.
desRemove	The number of times a DES remove was completed.
desEncodeGood	The number of packets successfully encrypted with DES.
desEncodeFail	The number of times DES encryption failed on a packet.
desEncodeDiscard	The number of packets discarded by DES encryption.
desEncodeGetInfoFail	The number of times no DES info was found for an encrypted packet.
desAttachGood	The number of times a DES attach was successful.
desConfigureGood	The number of times a DES configure was successful.
desDetached	The number of times a DES detach was completed.
desDecodeGood	The number of packets successfully decrypted using DES.
desDecodeFail	The number of times DES decryption failed on a packet.

Table 45-5: Parameters in the output of the **show ipsec counter=alg** command

Parameter	Meaning
desDecodeDiscard	The number of packets discarded by DES decryption.
TRIPLE DES INNER	Counter for the Triple DES Inner algorithm processing unit.
3DesInnerKeymatProcessed	The number of times Triple DES Inner key material has been processed.
3DesInnerAttachFail	The number of times a Triple DES Inner attach failed.
3DesInnerConfigureFail	The number of times a Triple DES Inner configure failed.
3DesInnerRemove	The number of times a Triple DES Inner remove was completed.
3DesInnerEncodeGood	The number of packets successfully encrypted with Triple DES Inner.
3DesInnerEncodeFail	The number of times Triple DES Inner encryption failed on a packet.
3DesInnerEncodeDiscard	The number of packets discarded by Triple DES Inner encryption.
3DesInnerEncGetInfoFail	The number of times no Triple DES Inner info was found for an encrypted packet.
3DesInnerAttachGood	The number of times a Triple DES Inner attach was successful.
3DesInnerConfigureGood	The number of times a Triple DES Inner configure was successful.
3DesInnerDetached	The number of times a Triple DES Inner detach was completed.
3DesInnerDecodeGood	The number of packets successfully decrypted with Triple DES Inner.
3DesInnerDecodeFail	The number of times Triple DES Inner decryption failed on a packet.
3DesInnerDecodeDiscard	The number of times a packet was discarded by Triple DES Inner decryption.
TRIPLE DES OUTER	Counter for the Triple DES Outer algorithm processing unit.
3DesOuterKeymatProcessed	The number of times Triple DES Outer key material has been processed.
3DesOuterAttachFail	The number of times a Triple DES Outer attach failed.
3DesOuterConfigureFail	The number of times a Triple DES Outer configure failed.
3DesOuterRemove	The number of times a Triple DES Outer remove was completed.
3DesOuterEncodeGood	The number of times a packet was successfully encrypted using Triple DES Outer.
3DesOuterEncodeFail	The number of times Triple DES Outer encryption failed on a packet.
3DesOuterEncodeDiscard	The number of packets discarded by Triple DES Outer encryption.
3DesOuterEncGetInfoFail	The number of times no Triple DES Outer info was found for an encrypted packet.
3DesOuterAttachGood	The number of times a Triple DES Outer attach was successful.

Table 45-5: Parameters in the output of the **show ipsec counter=alg** command

Parameter	Meaning
3DesOuterConfigureGood	The number of times a Triple DES Outer configure was successful.
3DesOuterDetached	The number of times a Triple DES Outer detach was completed.
3DesOuterDecodeGood	The number of packets successfully decrypted with Triple DES Outer.
3DesOuterDecodeFail	The number of times Triple DES Outer decryption failed on a packet.
3DesOuterDecodeDiscard	The number of packets discarded by Triple DES Outer decryption.
TRIPLE DES 2KEY	Counter for the Triple DES 2 Key algorithm processing unit.
3Des2KeyKeymatProcessed	The number of times Triple DES 2 Key key material has been processed.
3Des2KeyAttachFail	The number of times a Triple DES 2 Key attach failed.
3Des2KeyConfigureFail	The number of times a Triple DES 2 Key configure failed.
3Des2KeyRemove	The number of times a Triple DES 2 Key remove was completed.
3Des2KeyEncodeGood	The number of packets successfully encrypted with Triple DES 2 Key.
3Des2KeyEncodeFail	The number of times Triple DES 2 Key encryption failed on a packet.
3Des2KeyEncodeDiscard	The number of packets discarded by Triple DES 2 Key encryption.
3Des2KeyEncGetInfoFail	The number of times no Triple DES 2 Key info was found for an encrypted packet.
3Des2KeyAttachGood	The number of times a Triple DES 2 Key attach was successful.
3Des2KeyConfigureGood	The number of times a Triple DES 2 Key configure was successful.
3Des2KeyDetached	The number of times a Triple DES 2 Key detach was completed.
3Des2KeyDecodeGood	The number of packets successfully decrypted with Triple DES 2 Key.
3Des2KeyDecodeFail	The number of times Triple DES 2 Key decryption failed on a packet.
3Des2KeyDecodeDiscard	The number of packets discarded by Triple DES 2 Key decryption.
AES	Counters for the AEs processing unit.
aesKeymatProcessed	The number of times AES key material has been processed.
aesAttachFail	The number of times an AES attach failed.
aesConfigureFail	The number of times an AES configure failed.
aesRemove	The number of times an AES remove was completed.
aesEncodeGood	The number of packets successfully encrypted with AES.
aesEncodeFail	The number of times AES encryption failed on a packet.
aesEncodeDiscard	The number of packets discarded by AES encryption.

Table 45-5: Parameters in the output of the **show ipsec counter=alg** command

Parameter	Meaning
aesEncodeGetInfoFail	The number of times no AES information was found for an encrypted packet.
aesAttachGood	The number of times an AES attach was successful.
aesConfigureGood	The number of times an AES configure was good.
aesDetached	The number of times an AES detach was completed.
aesDecodeGood	The number of packets successfully decrypted using AES.
aesDecodeFail	The number of times AES decryption failed on a packet.
aesDecodeDiscard	The number of packets discarded by AES decryption.
SHA	Counter for the SHA algorithm processing unit.
shaKeymatProcessed	The number of times SHA key material has been processed.
shaAttachFail	The number of times an SHA attach failed.
shaConfigureFail	The number of times an SHA configure failed.
shaRemove	The number of times an SHA remove was completed.
shaEncodeGood	The number of packets successfully hashed with SHA.
shaEncodeFail	The number of times an SHA hash failed on a packet.
shaEncodeDiscard	The number of packets discarded during an SHA hash.
shaAttachGood	The number of times an SHA attach was successful.
shaConfigureGood	The number of times an SHA configure was successful.
shaDetached	The number of times an SHA detach was completed.
shaDecodeGood	The number of packets successfully hashed with SHA.
shaDecodeFail	The number of times an SHA hash failed on a packet.
shaDecodeDiscard	The number of packets discarded during an SHA hash.
MD5	Counter for the MD5 algorithm processing unit.
md5KeymatProcessed	The number of times MD5 key material has been processed.
md5AttachFail	The number of times an MD5 attach failed.
md5ConfigureFail	The number of times an MD5 configure failed.
md5Remove	The number of times an MD5 remove was completed.
md5EncodeGood	The number of packets successfully hashed with MD5.
md5EncodeFail	The number of times an MD5 hash failed on a packet.
md5EncodeDiscard	The number of packets discarded during an MD5 hash.
md5AttachGood	The number of times an MD5 attach was successful.
md5ConfigureGood	The number of times an MD5 configure was successful.
md5Detached	The number of times an MD5 detach was completed.
md5DecodeGood	The number of packets successfully hashed with MD5.
md5DecodeFail	The number of times an MD5 hash failed on a packet.
md5DecodeDiscard	The number of packets discarded during an MD5 hash.
DES-MAC	Counter for the DES MAC algorithm processing unit.
desmacKeymatProcessed	The number of times DES-MAC key material has been processed.
desmacAttachFail	The number of times a DES-MAC attach failed.
desmacConfigureFail	The number of times a DES-MAC configure failed.

Table 45-5: Parameters in the output of the **show ipsec counter=alg** command

Parameter	Meaning
desmacRemove	The number of times a DES-MAC remove was completed.
desmacEncodeGood	The number of packets successfully hashed with DES-MAC.
desmacEncodeFail	The number of times a DES-MAC hash failed on a packet.
desmacEncodeDiscard	The number of packets discarded during a DES-MAC hash.
desmacAttachGood	The number of times a DES-MAC attach was successful.
desmacConfigureGood	The number of times a DES-MAC configure was successful.
desmacDetached	The number of times a DES-MAC detach was completed.
desmacDecodeGood	The number of packets successfully hashed with DES-MAC.
desmacDecodeFail	The number of times a DES-MAC hash failed on a packet.
desmacDecodeDiscard	The number of packets discarded during a DES-MAC hash.
LZS	Counter for the LZS algorithm processing unit.
lzsKeymatProcessed	The number of times LZS key material has been processed.
lzsAttachFail	The number of times an LZS attach failed.
lzsConfigureFail	The number of times an LZS configure failed.
lzsRemove	The number of times an LZS remove was completed.
lzsEncodeGood	The number of packets successfully compressed using LZS.
lzsEncodeFail	The number of times LZS compression failed on a packet.
lzsEncodeDiscard	The number of packets discarded during an LZS compression.
lzsAttachGood	The number of times an LZS attach was successful.
lzsConfigureGood	The number of times an LZS configure was successful.
lzsDetached	The number of times an LZS detach was completed.
lzsDecodeGood	The number of packets successfully decompressed with LZS.
lzsDecodeFail	The number of times LZS decompression failed on a packet.
lzsDecodeDiscard	The number of packets discarded during an LZS decompression.
IPSec	Counter for the IPSec algorithm processing unit. Not available on AR410 series routers.
IPSecKeymatProcessed	The number of times IPSec key material has been processed. Not available on AR410 series routers.
lIPSecAttachFail	The number of times an IPSec attach failed. Not available on AR410 series routers.
IPSecConfigureFail	The number of times an IPSec configure failed. Not available on AR410 series routers.
IPSecRemove	The number of times an IPSec remove was completed. Not available on AR410 series routers.
IPSecEncodeGood	The number of packets successfully compressed using IPSec. Not available on AR410 series routers.
IPSecEncodeFail	The number of times IPSec compression failed on a packet. Not available on AR410 series routers.
IPSecEncodeDiscard	The number of packets discarded during an IPSec compression. Not available on AR410 series routers.

Table 45-5: Parameters in the output of the **show ipsec counter=alg** command

Parameter	Meaning
IPSecAttachGood	The number of times an IPSec attach was successful. Not available on AR410 series routers.
IPSecConfigureGood	The number of times an IPSec configure was successful. Not available on AR410 series routers.
lzsDetached	The number of times an IPSec detach was completed. Not available on AR410 series routers.
IPSecDecodeGood	The number of packets successfully decompressed with IPSec. Not available on AR410 series routers.
IPSecDecodeFail	The number of times IPSec decompression failed on a packet. Not available on AR410 series routers.
IPSecDecodeDiscard	The number of packets discarded during an IPSec decompression. Not available on AR410 series routers.

Figure 45-10: Example output from the **show ipsec counter=comp** command

```
COMP setup/remove counters:
  setupGetSaFailed     0     setupFailed           0
  setupDone            0
  removeGetSaFailed    0     removeNothingDone     0
  removeDone           0

COMP outbound processing counters:
  bufChainCopy         0     compTooSmall          0
  nonExpansionBackoff  0     dataExpansion         0
  compressionStart     0     compressionFailImm    0
  compDoneGetSaFail    0     compressionFail       0
  compDoneSaBadState   0     compressionGood       0

COMP inbound processing counters:
  bufChainCopy         0
  decompressionStart   0     decompressionFailImm  0
  decompDoneGetSaFail  0     decompressionFail     0
  decompDoneSaBadState 0     decompressionGood     0
```

Table 45-6: Parameters in the output of the **show ipsec counter=comp** command

Parameter	Meaning
COMP setup/remove counters	Counters for setting up and removing COMP SAs.
setupGetSaFailed	The number of attempts to setup an IPComp SA that were aborted because the SA no longer existed.
setupDone	The number of successful attempts to setup an IPComp SA.
removeGetSaFailed	The number of attempts to remove an IPComp SA that were aborted because the SA no longer existed.
removeDone	The number of successful attempts to remove an IPComp SA.
setupFailed	The number of failed attempts to set up an IPComp SA.
removeNothingDone	The number of attempts to remove an IPComp SA that required no action.

Table 45-6: Parameters in the output of the **show ipsec counter=comp** command

Parameter	Meaning
COMP outbound processing counters	Counters for the processing of outbound COMP SAs.
bufChainCopy	The number of outbound packets copied from a chain to a buffer prior to IPComp processing.
nonExpansionBackoff	The number of outbound packets not compressed due to a non-expansion backoff.
compressionStart	The number of times IPComp compression processing started on an outbound packet.
compDoneGetSaFail	The number of times an IPComp compression was aborted because the SA no longer existed.
compDoneSaBadState	The number of times an IPComp compression was aborted because the SA was no longer valid.
compTooSmall	The number of outbound packets not compressed because they were too small.
dataExpansion	The number of outbound packets expanded when they were compressed.
compressionFailImm	The number of times an IPComp compression failed immediately.
compressionFail	The number of times an IPComp compression failed.
compressionGood	The number of times an IPComp compression was completed successfully.
COMP inbound processing counters	Counters for processing inbound COMP SAs.
bufChainCopy	The number of inbound packets copied from a chain to a buffer prior to IPComp processing.
decompressionStart	The number of times IPComp decompression processing started on an inbound packet.
decompDoneGetSaFail	The number of times IPComp decompression was aborted because the SA no longer existed.
decompDoneSaBadState	The number of times IPComp decompression was aborted because the SA was no longer valid.
decompressionFailImm	The number of times IPComp decompression failed immediately.
decompressionFail	The number of times IPComp decompression failed.
decompressionGood	The number of times IPComp decompression completed successfully.

Figure 45-11: Example output from the **show ipsec counter=esp** command

```
ESP setup/remove counters

  setupGetSaFailed          0  setupEncSetupFailed       0
  setupHashSetupFailImm     0  setupEncSetupBundleRm     0
  setupFailed               0  setupDone                10
  removeGetSaFailed         0  removeNothingDone         0
  removeHashFailImm         0  removeDone                5

ESP outbound processing counters
  bufChainCopy              0  seqNumberCycled           0
  encryptionStart          17  encryptionFailImm         0
  encDoneGetSaFail          0  encryptionFail            0
  encDoneSaBadState         0  encryptionGood           17
  hashStart                 0  hashFailImm               0
  hashDoneGetSaFail         0  hashFail                  0
  hashDoneSaBadState        0  hashGood                  0

ESP inbound processing counters
  bufChainCopy              0  icvInvalid                0
  paddingInvalid            0  replayedPacket            0
  hashStart                 0  hashFailImm               0
  hashDoneGetSaFail         0  hashFail                  0
  hashDoneSaBadState        0  hashGood                  0
  decryptionStart          13  decryptionFailImm         0
  encDoneGetSaFail          0  decryptionFail            0
  encDoneSaBadState         0  decryptionGood           13
```

Table 45-7: Parameters in the output of the **show ipsec counter=esp** command

Parameter	Meaning
ESP setup/remove counters	Counters for setting up and removing ESP SAs.
setupGetSaFailed	The number of times an attempt to setup an ESP SA was aborted because the SA no longer existed.
setupHashSetupFailImm	The number of times an attempt to setup an ESP SA hash algorithm failed immediately.
setupFailed	The number of times an attempt to setup an ESP SA failed.
removeGetSaFailed	The number of times an attempt to remove an ESP SA was aborted because the SA no longer existed.
removeHashFailImm	The number of attempts to remove an ESP SA hash algorithm that failed immediately.
setupEncSetupFailed	The number of failed attempts to setup an ESP SA encryption algorithm.
setupEncSetupBundleRm	The number of attempts to setup an ESP SA that were aborted because the SA was being removed.
setupDone	The number of successful attempts to setup an ESP SA.
removeNothingDone	The number of attempts to remove an ESP SA that required no action.
removeDone	The number of successful attempts to setup an ESP SA.
ESP outbound processing counters	Counters for the processing of outbound ESP SAs.
bufChainCopy	The number of outbound packets copied from a chain to a buffer prior to ESP processing.

Table 45-7: Parameters in the output of the **show ipsec counter=esp** command

Parameter	Meaning
encryptionStart	The number of times ESP encryption processing started on an outbound packet.
encDoneGetSaFail	The number of times an ESP encryption was aborted because the SA no longer existed.
encDoneSaBadState	The number of times an ESP encryption was aborted because the SA was no longer valid.
hashStart	The number of times ESP hash processing started on an outbound packet.
hashDoneGetSaFail	The number of times an ESP hash was aborted because the SA no longer existed.
hashDoneSaBadState	The number of times an ESP hash was aborted because the SA was no longer valid.
seqNumberCycled	The number of times an outbound packet caused an ESP sequence number to cycle.
encryptionFailImm	The number of times ESP encryption failed immediately.
encryptionFail	The number of times ESP encryption failed.
encryptionGood	The number of times ESP encryption was completed successfully.
hashFailImm	The number of times an ESP hash failed immediately.
hashFail	The number of times an ESP hash failed.
hashGood	The number of times an ESP hash was completed successfully.
ESP inbound processing counters	Counters for the processing of inbound ESP SAs.
bufChainCopy	The number of inbound packets copied from a chain to a buffer prior to ESP processing.
paddingInvalid	The number of inbound ESP packets seen with invalid padding.
hashStart	The number of times ESP hash processing started on an inbound packet.
hashDoneGetSaFail	The number of times an ESP hash was aborted because the SA no longer existed.
hashDoneSaBadState	The number of times an ESP hash was aborted because the SA was no longer valid.
decryptionStart	The number of times ESP decryption processing started on an inbound packet.
encDoneGetSaFail	The number of times ESP decryption was aborted because the SA no longer existed.
encDoneSaBadState	The number of times ESP decryption was aborted because the SA was no longer valid.
icvInvalid	The number of inbound ESP packets seen with an invalid ICV.
replayedPacket	The number of inbound ESP packets seen with an invalid sequence number.
hashFailImm	The number of times an ESP hash failed immediately.
hashFail	The number of times an ESP hash failed.

Table 45-7: Parameters in the output of the **show ipsec counter=esp** command

Parameter	Meaning
hashGood	The number of times an ESP hash was completed successfully.
decryptionFailImm	The number of times ESP decryption failed immediately.
decryptionFail	The number of times ESP decryption failed.
decryptionGood	The number of times ESP decryption was completed successfully.

Figure 45-12: Example output from the **show ipsec counter=main** command

```
IPsec main packet processing counters:
  outProcessPkt           40    inProcessPkt             33
  outNoPolicyFound         0    inNoPolicyFound           0
  outProcessPktFinished   17    inProcessPktFinished     13
                                inProcessPktKeepalive   219


IPsec over UDP counters:
  outPkt                  10    inPkt                    10
  outPktFail               0    inPktBadVersion           0
                                inPktNoPolicy             0
  outUdpHeartBeat          0    in UdpHeartBeat           0
```

Table 45-8: Parameters in the output of the **show ipsec counter=main** command

Parameter	Meaning
IPsec main packet processing counters	
outProcessPkt	The number of outbound packets seen by IPsec.
outNoPolicyFound	The number of outbound packets seen by IPsec that did not match an IPsec policy.
outProcessPktFinished	The number of times IPsec processing was completed on an outbound packet.
inProcessPkt	The number of inbound packets seen by IPsec.
inNoPolicyFound	The number of inbound packets seen by IPsec that did not match an IPsec policy.
inProcessPktFinished	The number of times IPsec processing was completed on an inbound packet.
inProcessPktKeepalive	The number of NAT keepalive packets received.
IPsec over UDP counters	
outPkt	The number of outbound packets transmitted over UDP.
outPktFail	The number of failed attempts to tunnel an IPsec packet over UDP.
outUdpHeartBeat	The number of UDP heartbeats transmitted.
inPkt	The number of inbound packets received over UDP.
inPktBadVersion	The number of UDP tunnelled packets received with bad version numbers.
inPktNoPolicy	The number of UDP tunnelled packets received without a policy with UDP tunnelling enabled.

Table 45-8: Parameters in the output of the **show ipsec counter=main** command

Parameter	Meaning
inUdpHeartBeat	The number of UDP heartbeats received.

Figure 45-13: Example output from the **show ipsec counter=sad** command

```
SAD Counters
                      Good      Failed
  Find SA               13           0
  Get SA                64           1
  Add SA                10
  Delete SA              5           0
```

Table 45-9: Parameters in the output of the **show ipsec counter=sad** command

Parameter	Meaning
Good	The number of successful operations.
Failed	The number of unsuccessful operations.
Find SA	The number of successful and unsuccessful searches of the Security Association Database (SAD).
Get SA	The number of successful and unsuccessful attempts to access the Security Association Database (SAD).
Add SA	The number of successful attempts to add SAs to the Security Association Database (SAD).
Delete SA	The number of successful and unsuccessful attempts to delete SAs from the Security Association Database (SAD).

Figure 45-14: Example output from the **show ipsec counter=setup** command

```
IPsec bundle setup/remove counters:
  setupGetSaSpecFail        0   setupGetPolicyFail      0
  setupStarted             10   setupSaSetupFailImm     0
  setupSaSetupStarted      10   setupSaSetupFailed      0
  setupDone                10   setupFailed             0
  setupBundleRemoving       0
  removeStarted             5   removeSaSetupStarted    0
  removeDone                5
```

Table 45-10: Parameters in the output of the **show ipsec counter=setup** command

Parameter	Meaning
setupGetSaSpecFail	The number of SA specifications not found during an SA bundle setup.
setupStarted	The number of attempts to set up an SA bundle.
setupSaSetupStarted	The number of attempts to set up an SA.
setupDone	The number of SA bundles successfully set up.
setupBundleRemoving	The number of aborted attempts to set up an SA bundle because the bundle was being removed.
removeStarted	The number of attempts to remove an SA bundle.
removeDone	The number of SA bundles successfully removed.
setupGetPolicyFail	The number of attempts to set up an SA that failed because the IPsec policy was not found.
setupSaSetupFailImm	The number of attempts to set up an SA that failed immediately.
setupSaSetupFailed	The number of failed attempts to set up an SA.
setupFailed	The number of failed attempts to set up an SA bundle.
removeSaSetupStarted	The number of attempts to remove an SA.

Figure 45-15: Example output from the **show ipsec counter=spd** command

```
SPD Counters
                                Good          Failed
 policyMatchSelectors            59              17
 policyAdd                        2               0
 policyGet                       21               2
 policyDelete                     0               0
 policyGetConfig                  1               0
 policySetConfig                  1               0
 policyFindByPeer                 0               0
 saSpecAdd                        1               0
 saSpecGet                       21               1
 saSpecDelete                     0               0
 bundlespecAdd                    1               0
 bundleSpecGet                   12               1
 bundleSpecDelete                 0               0

Policy Filter Counters

 localAddressMaskFailed    0   localAddressRangeFailed    0
 remoteAddressMaskFailed   0   remoteAddressRangeFailed   0
 localPortFailed          17   remotePortFailed           0
 localNameFailed           0   remoteNameFailed           0
 transportProtoFailed      0
```

Table 45-11: Parameters in the output of the **show ipsec counter=spd** command

Parameter	Meaning
Good	The number of successful operations.
Failed	The number of unsuccessful operations.
policyMatchSelectors	The number of successful and unsuccessful attempts to search the Security Policy Database (SPD).
policyAdd	The number of successful and unsuccessful attempts to add a policy to the Security Policy Database (SPD).
policyGet	The number of successful and unsuccessful attempts to access a policy in the Security Policy Database (SPD).
policyDelete	The number of successful and unsuccessful attempts to delete a policy from the Security Policy Database (SPD).
policyGetConfig	The number of successful and unsuccessful attempts to convert policy to policy configuration format when modifying policy parameters.
policySetConfig	The number of successful and unsuccessful attempts to convert policy configuration format to policy when modifying policy parameters.
policyFindByPeer	The number of successful and unsuccessful attempts search the Security Policy Database (SPD) by peer IP address.
saSpecAdd	The number of successful and unsuccessful attempts to add an SA specification to the Security Policy Database (SPD).
saSpecGet	The number of successful and unsuccessful attempts to access an SA specification in the Security Policy Database (SPD).
saSpecDelete	The number of successful and unsuccessful attempts to delete an SA specification from the Security Policy Database (SPD).
bundleSpecAdd	The number of successful and unsuccessful attempts to add a bundle specification to the Security Policy Database (SPD).
bundleSpecGet	The number of successful and unsuccessful attempts to access a bundle specifications in the Security Policy Database (SPD).
bundleSpecDelete	The number of successful and unsuccessful attempts to delete a bundle specification from the Security Policy Database (SPD).
localAddressMaskFailed	The number of failures trying to match the local IP address an mask.
remoteAddressMaskFailed	The number of failures trying to match the remote IP address and mask.
localPortFailed	The number of failures trying to match the local port selector field.
localNameFailed	The number of failures trying to match the local name selector field.
transportProtoFailed	The number of failures trying to match the transport protocol field.
localAddressRangeFailed	The number of failures trying to match the local IP address range.

Table 45-11: Parameters in the output of the **show ipsec counter=spd** command

Parameter	Meaning
remoteAddressRangeFailed	The number of failures trying to match the remote IP address range.
remotePortFailed	The number of failures trying to match the remote port selector field.
remoteNameFailed	The number of failures trying to match the remote name selector field.

Examples

To display all IPsec counters, use the command:

```
show ipsec counter
```

To display **spd** and **main** IPsec counter types, use the command:

```
show ipsec counter=spd,main
```

Related Commands

show ipsec sa

show ipsec policy

Syntax

```
SHow IPSec POLIcy[=name] [SABundle] [COUnter]
```

where *name* is a string 1 to 23 characters long. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.

Description

This command displays information about IPsec policies and SA bundles attached to IPsec policies.

The **policy** parameter specifies the name of the policy to display. A policy with the specified name must already exist. If a value is not specified, summary information about all policies displayed (Figure 45-16 on page 45-105, Table 45-12 on page 45-106). If a value is specified, detailed information about the specified policy is displayed (Figure 45-17 on page 45-106, Table 45-13 on page 45-107).

If **sabundle** is specified, only information about existing SA bundles belonging to the policy or policies is displayed (Figure 45-18 on page 45-109, Table 45-14 on page 45-109).

If **counter** is specified, counters for the specified policy or policies are displayed (Figure 45-19 on page 45-110, Table 45-15 on page 45-110).

Figure 45-16: Example output from the **show ipsec policy** command

```
Interface Name              Action  KeyManagement Position
-----------------------------------------------------------
PPP0      MY_PPP_POLICY1    IPSEC   ISAKMP          1
PPP0      MY_PPP_POLICY2    IPSEC   ISAKMP          2
PPP0      MY_PPP_POLICY3    DENY    -               3
```

Table 45-12: Parameters in the output of the **show ipsec policy** command

Parameter	Meaning
Interface	A interface name and logical index.
Name	A character string to identify the policy.
Action	Whether the required action is IPSEC, permit, or deny.
keymanagement	Whether the key management method is manual or ISAKMP.
Position	A number specifying the position of the policy.

Figure 45-17: Example output from the **show ipsec policy** command for a specific policy.

```
IPsec Policy Information

Name ........................... my_vpn
Interface ...................... PPP0
Source Interface ............... PPP0
Position ....................... 1
Action ......................... IPSEC

Key Management ................. ISAKMP
Isakmp Policy Name ............. my_isakmp_policy
Bundle Specification ........... 2
Peer IP Address Dynamic ........ FALSE
Peer IP address Any ............ FALSE
Local IP Address Dynamic ....... FALSE
Peer IP Address ................ 192.168.10.1
Local IP Address ............... 232.163.2.3
Use PFS Key .................... TRUE
Group .......................... 1
Filter:
  Local Address ................ 192.167.2.0
  Local Mask ................... 255.255.255.255
  Local Port ................... ANY
  Local Name ................... main_office
  Remote Address ............... 232.163.2.20
  Remote Mask .................. 255.255.255.255
  Remote Port .................. ANY
  Remote Name .................. ANY
  Transport Protocol ........... ANY
  SA Selector from Packet ...... 00000000

DF Bit ......................... COPY
UDP Tunnel ..................... TRUE
  Peer Port .................... 14997
  Peer IP Address .............. 202.36.163.204
  Internal IP Address .......... 192.168.1.1
  HeartBeats Enabled ........... FALSE
Debug device ................... 16
Filter debug flags ............. 00000000
Packet debug flags ............. 00000000
Trace debug flags .............. 00000000
Packet debug length ............ 72
Max Out Packet queue length .... 20
Number of Out Packets queued ... 0

Bundles
Bundle                  Expiry Limits - hard/soft/used
Index  SAs       State   Bytes                       Seconds
0      1,2,5     VALID   1000000/900000/907835       36000/30000/32432
1      12,13,14  INVALID 1000000/900000/0            36000/30000/0
```

Table 45-13: Parameters in the output of the **show ipsec policy** command for a specific policy.

Parameter	Meaning
Name	The manager-assigned name of the policy.
Interface	The logical interface to which the policy is attached.
Position	The position of this policy in the list of policies attached to the interface.
Action	Whether the action for this policy is IPsec, permit, or deny.
Key Manage	Whether the key management method for this policy is MANUAL or ISAKMP.
Isakmp Policy Name	The name of the ISAKMP policy name to be used in phase 1 negotiations.
Bundle Specification	The identification number of the SA bundle specification to be used by this policy.
Peer IP Address Dynamic	Whether the peer IP address is dynamically assigned.
Peer IP Address Any	Whether the policy can be used to negotiate IPsec SAs for any peer.
Local IP Address Dynamic	Whether the policy is attached to a dynamic IP interface.
Peer IP Address	The IP address of the remote end of the IPsec tunnel.
IP Route Template	The name of the IP route template this policy uses
Local IP Address	The IP address of the local end of the IPsec tunnel
Use PFS Key	Whether to use Perfect Forward Security (PFS).
Group	The Diffie-Hellman group to be used.
Filter	Information about the packet matching filter, or selectors, for this policy.
Local Address	The local address packet selector field for the policy. For IPv6 addresses, slash notation may be used to indicate the prefix length.
Local Mask	The local mask packet selector field for the policy, for IPv4.
Local Port	The local port packet selector field for the policy.
Local Name	The local name packet selector field for the policy.
Remote Address	The remote IP address packet selector field for the policy. For IPv6 addresses, slash notation may be used to indicate the prefix length.
Remote Mask	The remote mask packet selector field for the policy, for IPv4.
Remote Port	The remote port packet selector field for the policy.
Remote Name	The remote name packet selector field for the policy.
Transport Protocol	The transport protocol packet selector field for the policy.
SA Selector From Pkt	A flag specifying which packet selector fields from the packet are used to find a matching SA.
DF Bit	The value to be set for the Don't Fragment bit in the outer IP header. either COPY, SET, or CLEAR
UDP Tunnel	The status of UDP tunnelling for this policy; either TRUE (enabled) or FALSE (disabled).
Peer Port	The port to which UDP tunnelled traffic is sent.

Table 45-13: Parameters in the output of the **show ipsec policy** command for a specific policy. (continued)

Parameter	Meaning
Peer IP Address	The IP address to which UDP tunnelled traffic is sent.
Internal IP Address	The IP address to be used for the destination of the IPsec tunnelled packets.
HeartBeats Enabled	Whether UDP heartbeat mode is enabled or disabled.
Debug device	The device (port number) to which debugging output is sent.
Filter debug flags	A flag indicating the policy selectors and pass/fail results for which filter debugging is enabled.
Packet debug flags	A flag indicating the processing units and directions for which packet debugging is enabled, and the portions of packets to be displayed.
Trace debug flags	A flag indicating whether trace debugging is enabled.
Packet debug length	The length of the packet displayed for debugging.
Max Out Packet queue length	The maximum number of packets that can be queued before processing.
Number of Out Packets queued	The number of packets currently queued for processing.
Bundles	Information about the SA bundles attached to this policy.
Index	The identification number of an SA bundle attached to this policy.
SAs	The SA identification numbers of the SAs in the SA bundle.
State	The state of the SA bundle; either VALID, INVALID, CREATING, or REMOVING.
Expiry Limits - hard/soft/used	Information about the limits used to expire SAs created from this bundle. When a soft limit is exceed the SAs are renegotiated. When a hard limit is exceeded the SAs are removed from the Security Policy Database (SPD).
ExpiryBytes	The expiry information in bytes for each SA bundle. The first figure displays the hard expiry limit, the second figure displays the soft expiry limit and the third figure displays the number of bytes already used.
ExpirySeconds	The expiry information in seconds for each SA bundle. The first figure displays the hard expiry limit, the second figure displays the soft expiry limit and the third figure displays the seconds already used.

Figure 45-18: Example output from the **show ipsec policy sabundle** command

```
IPsec Policy SA Bundles

                         Expiry Limits - hard/soft/used
Index  SAs       State   Bytes                        Seconds

Policy ..... NY_OFFICE_GEN
0      1,2,5     VALID   1000000/900000/807835       36000/35500/35632
1      12,13,14  CREATING1000000/900000/0            36000/35500/0

Policy ..... NZ_OFFICE_GEN
0      3,4,7     VALID   100000/90000/93783          18000/17500/9432
1      9,10,11   VALID   100000/90000/1274           18000/17500/0

Policy ..... NY_FINANCE
0      15,16     VALID   10000/9000/2384             3600/3000/987
```

Table 45-14: Parameters in the output of the **show ipsec policy sabundle** command

Parameter	Meaning
Policy	The manager-assigned name of an IPsec policy.
Index	The identification number of an SA bundle attached to the policy.
SAs	The SA identification numbers of the SAs in the SA bundle.
State	The state of the SA bundle; either VALID, INVALID, CREATING, or REMOVING.
Expiry Limits - hard/soft/used	Information about the limits used to expire SAs created from this bundle. When a soft limit is exceed the SAs are renegotiated. When a hard limit is exceeded the SAs are removed from the Security Policy Database (SPD).
ExpiryBytes	The expiry information in bytes for each SA bundle. The first figure displays the hard expiry limit, the second figure displays the soft expiry limit and the third figure displays the number of bytes already used.
ExpirySeconds	The expiry information in seconds for each SA bundle. The first figure displays the hard expiry limit, the second figure displays the soft expiry limit and the third figure displays seconds already used.

Figure 45-19: Example output from the **show ipsec policy counter** command

```
Setup/Remove Counters:
  setupStarted                1   setupSaSetupFailImm         0
  setupSaSetupStarted         1   setupSaSetupFailed          0
  setupDone                   1   setupFailed                 0
  removeStarted               0   removeSaSetupStarted        0
  removeDone                  0

Outbound Packet Processing Counters:
  outDeny                     0   outPermit                   0
  outNoBundle                 1   outNoBundleFail             0
  outMakeSetupStrctFail       0   outSetupBundleFail          0
  outBundleSoftExpire         0   outBundleExpire             0
  outProcessStart          4373   outProcessFailImm           0
  outBundleStateBad           0   outProcessFail              0
  outProcessDone           4373

Inbound Packet Processing Counters:
  inDeny                      0   inPermit                    0
  inCompUncompressed          0   inActionIpsecFail           0
  inBundleStateBad            0   inNotFirstSaInBundle        0
  inProcessStart           4373   inProcessFailImm            0
  inProcessFail               0   inProcessDone            4373
  inEndOfBundle               0   inPrematureEndBundle        0
  inBundleSaMatchFail         0   inPolicyActionFail          0
  inPolSelectMatchFail        0   inBundleReplaced            0
  inBundleSoftExpire          0   inBundleExpire              0
```

Table 45-15: Parameters in the output of the **show ipsec policy counter** command

Parameter	Meaning
Setup/Remove Counters	Counters for creating and destroying SA bundles.
setupStarted	The number of attempts to set up an SA bundle.
setupSaSetupStarted	The number of attempts to set up an SA.
setupDone	The number of successful attempts to set up an SA bundle.
removeStarted	The number of attempts to remove an SA bundle.
removeDone	The number of SA bundles completed removed.
setupSaSetupFaillmm	The number of attempts to set up an SA that failed immediately.
setupSaSetupFailed	The number of failed attempts to set up an SA.
setupFailed	The number of failed attempts to set up an SA bundle.
removeSaSetupStarted	The number of attempts to remove an SA.
Outbound Packet Processing Counters	Counters for processing outbound SA bundles.
outDeny	The number of outbound packets that matched an IPsec policy with **deny** action.
outNoBundle	The number of outbound packets that matched an IPsec policy with no bundles.
outMakeSetupStrctFail	The number of attempts to setup an SA bundle for an outbound packet that failed because the setup structure failed.
outBundleSoftExpire	The number of times an outbound packet caused the soft expiry kilobyte limit of an SA bundle to be reached.

Table 45-15: Parameters in the output of the **show ipsec policy counter** command

Parameter	Meaning
outProcessStart	The number of times IPsec processing started on an outbound packet.
outBundleStateBad	The number of outbound packets that matched an IPsec policy with no valid bundles.
outProcessDone	The number of times IPsec processing finished successfully on an outbound packet.
outPermit	The number of outbound packets that matched an IPsec policy with PERMIT action.
outNoBundleFail	The number of outbound packets failed by IPsec because they matched an IPsec policy with no bundles.
outSetupBundleFail	The number of failed attempts to setup an SA bundle for an outbound packet.
outBundleExpire	The number of times an outbound packet caused the expiry kilobyte limit of an SA bundle to be reached.
outProcessFailImm	The number of times IPsec processing failed immediately on an outbound packet.
outProcessFail	The number of times IPsec processing failed on an outbound packet.
Inbound Packet Processing Counters	Counters for processing inbound SA bundles.
inDeny	The number of inbound packets that matched an IPsec policy with DENY action.
inCompUncompressed	The number of uncompressed inbound packet seen on an IPComp SA.
inBundleStateBad	The number of inbound packets that matched an IPsec policy with no valid bundles.
inProcessStart	The number of times IPsec processing started on an inbound packet.
inProcessFail	The number of times IPsec processing failed on an inbound packet.
inEndOfBundle	The number of inbound packets not completely processed by the chosen SA bundle.
inBundleSaMatchFail	The number of inbound packets that did not match an SA in the chosen SA bundle.
inPolSelectMatchFail	The number of inbound packets that did not match the selectors of the IPsec policy by which it was processed.
inBundleSoftExpire	The number of times an inbound packet caused the soft expiry kilobyte limit of an SA bundle to be reached.
inPermit	The number of inbound packets that matched an IPsec policy with PERMIT action.
inActionIpsecFail	The number of plaintext inbound packets that matched an IPsec policy with IPSEC action.
inNotFirstSaInBundle	The number of inbound packets that matched an SA that was not the first SA in a bundle.
inProcessFailImm	The number of times IPsec processing failed immediately on an inbound packet.

Table 45-15: Parameters in the output of the **show ipsec policy counter** command

Parameter	Meaning
inProcessDone	The number of times IPsec processing finished successfully on an inbound packet.
inPrematureEndBundle	The number of inbound packets completely processed before all the SAs in the chosen SA bundle were used.
inPolicyActionFail	The number of inbound IPsec packets seen that did not match an IPsec policy.
inBundleReplaced	The number of inbound packets that removed an obsolete SA bundle.
inBundleExpire	The number of times an inbound packet caused the soft expiry kilobyte limit of an SA bundle to be reached.

Examples

To display a policy with the name "my_vpn", use the command:

```
show ipsec policy="my_vpn"
```

Related Commands

create ipsec policy
destroy ipsec policy
set ipsec policy

show ipsec sa

Syntax `SHow IPSec SA[=sa-id] [COUnter]`

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

Description This command displays information about the specified SA or all SAs in the Security Association Database (SAD).

The **sa** parameter specifies the SA for which detailed information is to be displayed. If a value is not specified, summary information about all SAs is displayed (Figure 45-20 on page 45-113, Table 45-16 on page 45-113). If a value is specified, detailed information about the specified SA is displayed (Figure 45-21 on page 45-114, Table 45-17 on page 45-115).

If **counter** is specified, counters for the specific SA are displayed (Figure 45-22 on page 45-117, Table 45-18 on page 45-117).

Figure 45-20: Example output from the **show ipsec sa** command

```
SA Id  Policy                Bundle  State    Protocol    OutSPI       InSPI
-------------------------------------------------------------------------------
    5  MY_VPN1_POLICY             5  Valid    ESP       554781384  2776920516
    6  MY_VPN1_POLICY             4  Valid    AH       2684492123  2115024437
    7  MY_VPN1_POLICY             3  Valid    ESP       504032446   987203183
    8  MY_VPN1_POLICY             2  Valid    ESP       736637299  1499049187
    9  MY_VPN1_POLICY             1  Valid    AH       2387532688    89042001
```

Table 45-16: Parameters in the output of the **show ipsec sa** command

Parameter	Meaning
SA Id	The identification number for the SA.
Policy	The name of the IPsec policy to which the SA is attached.
Bundle	The bundle identification number of the bundle attached to the policy.
State	The state of the SA; either UNDEF, VALID, CREATING, or REMOVING.
Protocol	The name of the IPsec protocol used by the SA; either ESP, AH, or IPComp.
OutSPI	The Security Parameter Index (SPI) for the outbound traffic.
InSPI	The Security Parameter Index (SPI) for the inbound traffic.

Figure 45-21: Example output from the **show ipsec sa** command for a specific SA.

```
SA Id ................................ 2
 Policy ............................... roaming1
 Bundle ............................... 1
 SA Specification Used ................ 1
 State ................................ Valid
 Protocol ............................. ESP
 Role ................................. RESPONDER
 Mode ................................. UDP_ENCAP_TRANSPORT
 Outbound SPI ......................... 736637299
 Inbound SPI .......................... 1499049187
 Encryption algorithm ................. DES
 Encryption ENCO channel............... 1
 Hash algorithm ....................... SHA
 Hash ENCO channel..................... 2
 NAT Traversal NAT-OA
  Peer original source IP address ..... 192.168.1.10
  Peer original destination IP address.. -
 Filters
  Local IP address ..................... 10.10.10.2
  Local IP address mask ................ 255.255.255.255
  Remote IP address .................... 10.10.10.1
  Remote IP address mask ............... 255.255.255.255
  Local port number .................... 1701
  Remote port number ................... 1701
  NAPT remote port number .............. 2
  Transport protocol ................... UDP
  Local Name ........................... ANY
  Remote Name .......................... atr02
 DF BIT ............................... CLEAR
 Last sent sequence number ............ 27
 Anti-replay checking enabled ......... TRUE
 Anti-replay window size .............. 32
 Last received sequence number ......... 5
 Anti-replay bitmap ................... 000000ff
 Debug device ......................... 16
 Filter debug flags ................... 000000ff
 Packet debug flags ................... 00000067
 Trace debug flags .................... 00000001
 Packet debug length .................. 72
```

Table 45-17: Parameters in the output of the **show ipsec sa** command for a specific SA.

Parameter	Meaning
SA Id	The identification number for the SA.
Policy	The name of the IPsec policy to which this SA is attached.
Bundle	The identification number for the bundle attached to the policy.
SA Specification Used	The identification number of the SA specification used to create this SA.
State	Whether the current state of the SA is UNDEF, VALID, CREATING, or REMOVING.
Protocol	The IPsec protocol used by this SA is ESP, AH, or COMP.
CPI	The Compression Parameter Index. Displayed when Protocol is set to COMP.
Role	Whether this peer acted as the initiator or responder in order to create this SA.
Mode	The IPsec operational mode for this SA: TUNNEL TRANSPORT UDP_ENCAPSULATED_TUNNEL UDP_ENCAPSULATED_TRANSPORT
Outbound SPI	The Security Parameter Index (SPI) for outbound traffic.
Inbound SPI	The Security Parameter Index (SPI) for inbound traffic.
Encryption algorithm	The encryption algorithm used by this SA. Displayed when Protocol is set to ESP.
Encryption ENCO channel	The ENCO channel number for the encryption process. Displayed when Protocol is set to ESP.
Hash algorithm	The hash algorithm used by this SA. Displayed when Protocol is set to ESP or AH.
Hash ENCO channel	The ENCO channel number for the hash process. Displayed when Protocol is set to ESP or AH.
Compression algorithm	The compression algorithm used by this SA. Displayed when Protocol is set to COMP.
Compression ENCO channel	The ENCO channel number for the compression process. Displayed when Protocol is set to COMP.
NAT-Traversal NAT-OA	Information about original IP addresses.
Peer original source IP address	Source IP address that the remote peer uses when sending packets to this peer. UDP-encapsulated transport mode only.
Peer original destination IP address	Destination IP address that the remote peer uses when sending packets to this peer. UDP-encapsulated transport mode, and IETF draft v08, *Negotiation of NAT-T in the IKE*.
Filters	Information about the packet selectors for this SA.
Local IP address	The local IP address packet selector field of the policy to which this SA is attached.
Local IP address mask	The local IP address mask packet selector field of the policy to which this SA is attached.
Remote IP address	The remote IP address packet selector field of the policy to which this SA is attached.
Remote IP address mask	The remote IP address mask packet selector field of the policy to which this SA is attached.

Table 45-17: Parameters in the output of the **show ipsec sa** command for a specific SA.

Parameter	Meaning
Local port number	The local IP port number packet selector field of the policy to which this SA is attached.
Remote port number	The remote IP port number packet selector field of the policy to which this SA is attached.
NAPT remote port number	Network Address Port Translation number. Multiple clients in UDP-encapsulated transport mode appear to come from the same source. Therefore, NAT-T changes the source port to this value to maintain a distinction.
Transport protocol	The transport protocol packet selector field of the policy to which this SA is attached.
Local Name	The local name packet selector field of the policy to which this SA is attached.
Remote Name	The remote name packet selector field of the policy to which this SA is attached.
DF Bit	The value specified for the DF (Don't Fragment) bit in the DFBIT parameter of the policy to which this SA is attached. Either COPY, CLEAR, or SET.
Last sent sequence number	The sequence number of the last packet sent by this SA.
Anti-replay checking enabled	Whether anti-replay checking is enabled.
Anti-replay window size	The window size for anti-replay checking.
Last received sequence number	The sequence number of the last packet received by this SA.
Anti-replay bitmap	The bitmap of the anti-replay window. The bitmap is *n* bits long, where *n* is the anti-replay window size, and records the last *n* sequence numbers, up to and including the last sequence number received. A "1" means a packet with that sequence number has been received, and a "0" means a packet with that sequence number has not been received.
Debug device	The device (port number) to which debugging output is sent.
Filter debug flags	A flag indicating the policy selectors and pass/fail results for which filter debugging is enabled.
Packet debug flags	A flag indicating the processing units and directions for which packet debugging is enabled, and the portions of packets to be displayed.
Trace debug flags	A flag indicating whether trace debugging is enabled.
Packet debug length	The data length of the packet displayed for packet debugging.

Figure 45-22: Example output from the show **ipsec sa counter** command

```
SA: 8

Outbound packet processing counters:
  outProcessStart       0    outProcessFailImm     0
  outProcessFail        0    outProcessDone        0
ESP:
  espOutBufChainCopy    0
  espEncryptionStart    0    espEncryptionFailImm  0
  espEncryptionFail     0    espEncryptionGood     0
  espOutHashStart       0    espOutHashFailImm     0
  espOutHashFail        0    espOutHashGood        0
AH:
  ahOutBufChainCopy     0    ahSeqNumberCycled     0
  ahOutHashStart        0    ahOutHashFailImm      0
  ahOutHashFail         0    ahOutHashGood         0
IPCOMP:
  compOutBufChainCopy   0    compTooSmall          0
  nonExpansionBackoff   0    dataExpansion         0
  compressionStart      0    compressionFailImm    0
  compressionFail       0    compressionGood       0

Inbound packet processing counters:
  inProcessStart        0    inProcessFailImm      0
  inProcessFail         0    inProcessDone         0
ESP:
  espInBufChainCopy     0    espReplayedPacket     0
  espInHashStart        0    espInHashFailImm      0
  espInHashFail         0    espInHashGood         0
  espDecryptionStart    0    espDecryptionFailImm  0
  espDecryptionFail     0    espDecryptionGood     0
  espIcvInvalid         0    espPaddingInvalid     0
AH:
  ahInBufChainCopy      0    ahReplayedPacket      0
  ahBadPayloadLength    0    ahIcvInvalid          0
  ahInHashStart         0    ahInHashFailImm       0
  ahInHashFail          0    ahInHashGood          0
IPCOMP:
  compInBufChainCopy    0
  decompressionStart    0    decompressionFailImm  0
  decompressionFail     0    decompressionGood     0
```

Table 45-18: Parameters in the output of the **show ipsec sa counter** command

Parameter	Meaning
outProcessStart	The number of times processing started on an outbound packet.
outProcessFail	The number of times outbound packet processing failed.
outProcessFailImm	The number of times outbound packet processing failed immediately.
outProcessDone	The number of times outbound packet processing successfully completed.
espOutBufChainCopy	The number of outbound packets copied from a chain to a buffer prior to ESP processing.
espEncryptionStart	The number of times ESP encryption started.
espEncryptionFail	The number of times ESP encryption failed.

Table 45-18: Parameters in the output of the **show ipsec sa counter** command

Parameter	Meaning
espOutHashStart	The number of times ESP authentication started.
espOutHashFail	The number of times ESP authentication failed.
espEncryptionFailImm	The number of times ESP encryption failed immediately.
espEncryptionGood	The number of times ESP encryption successfully completed.
espOutHashFailImm	The number of times ESP authentication failed immediately.
espOutHashGood	The number of times ESP authentication successfully completed.
ahOutBufChainCopy	The number of outbound packets copied from a chain to a buffer prior to AH processing.
ahOutHashStart	The number of times AH authentication started.
ahOutHashFail	The number of times AH authentication failed.
ahSeqNumberCycled	The number of times the AH sequence number has cycled.
ahOutHashFailImm	The number of times AH authentication failed immediately.
ahOutHashGood	The number of times AH authentication successfully completed.
compOutBufChainCopy	The number of outbound packets copied from a chain to a buffer prior to IPComp processing.
nonExpansionBackoff	The number of packets not compressed due to a non-expansion backoff.
compressionStart	The number of times IPComp compression started.
compressionFail	The number of times IPComp compression failed.
compTooSmall	The number of packets not compressed because they were too small.
dataExpansion	The number of packets expanded when they were compressed.
compressionFailImm	The number of times IPComp compression failed immediately.
compressionGood	The number of times IPComp compression completed successfully.
inProcessStart	The number of times processing started on an inbound packet.
inProcessFail	The number of times inbound packet processing failed.
inProcessFailImm	The number of times inbound packet processing failed immediately.
inProcessDone	The number of times inbound packet processing successfully completed.
espInBufChainCopy	The number of inbound packets copied from a chain to a buffer prior to ESP processing.
espInHashStart	The number of times ESP authentication started.
espInHashFail	The number of times ESP authentication failed.
espDecryptionStart	The number of times ESP decryption started.
espDecryptionFail	The number of times ESP decryption failed.
espIcvInvalid	The number of ESP packets seen with an invalid ICV.

Table 45-18: Parameters in the output of the **show ipsec sa counter** command

Parameter	Meaning
espReplayedPacket	The number of ESP packets seen with an invalid sequence number.
espInHashFailImm	The number of times ESP authentication failed immediately.
espInHashGood	The number of times ESP authentication successfully completed.
espDecryptionFailImm	The number of times ESP decryption failed immediately.
espDecryptionGood	The number of times ESP decryption successfully completed.
espPaddingInvalid	The number of ESP packets seen with invalid padding.
ahInBufChainCopy	The number of inbound packets copied from a chain to a buffer prior to AH processing.
ahBadPayloadLength	The number of AH packets seen with an invalid payload length.
ahInHashStart	The number of times AH authentication started.
ahInHashFail	The number of times AH authentication failed.
ahReplayedPacket	The number of AH packets seen with an invalid sequence number.
ahIcvInvalid	The number of AH packets seen with an invalid ICV.
ahInHashFailImm	The number of times AH authentication failed immediately.
ahInHashGood	The number of times AH authentication successfully completed.
compInBufChainCopy	The number of inbound packets copied from a chain to a buffer prior to IPComp processing.
decompressionStart	The number of times IPComp decompression started.
decompressionFail	The number of times IPComp decompression failed.
decompressionFailImm	The number of times IPComp decompression failed immediately.
decompressionGood	The number of times IPComp decompression completed successfully.

Examples To display a list of all SA entries in the IPsec Security Association Database, use the command:

```
sh ips sa
```

To display information about Security Association 1 in the IPsec Security Association Database, use the command:

```
sh ips sa=1
```

To display counters for Security Association 1 in the IPsec Security Association Database, use the command:

```
sh ips sa=1 cou
```

Related Commands **show ipsec**

show ipsec saspecification

Syntax `SHow IPSec SASpecification[=`*`spec-id`*`]`

where *spec-id* is a integer from 0 to 255

Description This command displays information about SA specifications.

The **saspecification** parameter specifies the SA specification is to be displayed. If a value is not specified, summary information about all SA specifications is displayed (Figure 45-23 on page 45-120, Table 45-19 on page 45-120). If a value is specified, detailed information about the specified SA specification is displayed (Figure 45-24 on page 45-121, Table 45-20 on page 45-121).

Figure 45-23: Example output from the **show ipsec saspecification** command

```
SA Specifications

ID Proto  KeyMan  Mode       EncAlg      HashAlg CompAlg InSpi  EncKey HashKey
-------------------------------------------------------------------------------
 1 ESP    MANUAL  TUNNEL     3DESINNER   NULL    -         300       5       -
 2 AH     MANUAL  TUNNEL     NULL        SHA     -         301       -       6
 3 ESP    MANUAL  TUNNEL     3DES2KEY    MD5     -         302       7       8
 4 ESP    ISAKMP  TRANSP     DES         SHA     -           -       -       -
 5 COMP   ISAKMP  TUNNEL     NULL        NULL    LZS         -       -       -
```

Table 45-19: Parameters in the output of the **show ipsec saspecification** command

Parameter	Meaning
ID	The identification number for the SA specification.
Protocol	Whether the IPsec protocol used by the SA specification is ESP, AH, or IPComp.
KeyMan	Whether the key management method used by the SA specification is manual or ISAKMP.
Mode	Whether the encapsulation mode used by the SA specification is tunnel or transport.
EncAlg	The encryption algorithm used by the SA specification; either AES128, AES192, AES256, DES, 3DES2KEY, 3DESINNER, 3DESOUTER, or NULL.
HashAlg	The hash algorithm used by the SA specification; either MD5, SHA, DESMAC, or NULL.
CompAlg	The compression algorithm used by the SA specification; only LZS is supported.
InSpi	The Security Parameter Index (SPI) for inbound traffic.
EncKey	The identification number of the ENCO key to be used for the encryption algorithm.
HashKey	The identification number of the ENCO key to be used for the hash algorithm.

Figure 45-24: Example output from the **show ipsec saspecification** command for a specific SA specification.

```
SA Specification
Id ..................... 1
Protocol ............... ESP
Key Management ......... MANUAL
Mode ................... TUNNEL
Encryption Algorithm..... 3DES2KEY
Hash Algorithm ......... NULL
Compression Algorithm ... -
In SPI ................. 300
Out SPI ................ 400
Encryption Key ......... 5
Hash Key ............... -
AntiReplay Enabled ..... -
Replay Window Size ..... -
```

Table 45-20: Parameters in the output of the **show ipsec saspecification** command for a specific SA specification.

Parameter	Meaning
ID	The number used to identify the Sa specification.
Protocol	Whether the IPsec protocol used by the SA specification is ESP, AH, or IPComp.
Key Management	Whether the key management method used by the SA specification is manual or ISAKMP.
Mode	Whether the encapsulation mode used by the SA specification is tunnel or transport.
Encryption Algorithm	The encryption algorithm used by the SA specification; either AES128, AES192, AES256, DES, 3DES2KEY, 3DESINNER, 3DESOUTER, or NULL.
Hash Algorithm	Whether the hash algorithm used by the SA specification is MD5, SHA, DESMAC, or NULL.
Compression Algorithm	The compression algorithm used by the SA specification; Only LZS is supported.
In SPI	The Security Parameter Index (SPI) for inbound traffic.
Out SPI	The Security Parameter Index (SPI) for outbound traffic.
Encryption Key	The identification number of the ENCO key to be used for the encryption algorithm.
Hash Key	The identification number of the ENCO key to be used for the hash algorithm.
AntiReplay Enabled	Whether the anti-replay mechanism is enabled.
Replay Window Size	The size of the anti replay window. Possible values are 32,64, 128, or 256.

Examples To display all SA specifications, use the command:

```
sh ips sas
```

To display an SA proposal with proposal id 1, use the command:

```
sh ips sas=1
```

Related Commands

create ipsec saspecification
destroy ipsec saspecification
set ipsec saspecification

show isakmp

Syntax

```
SHow ISAkmp
```

Description

This command shows the status of ISAKMP (Figure 45-25 on page 45-122, Table 45-21 on page 45-122).

Figure 45-25: Example output from the **show isakmp** command

```
ISAKMP Module Configuration

  Module Status ......................... DISABLED
  UDP Port .............................. 500
  Debug Flag ............................ 00000000
  Global Debug Device ................... 0
  Local RSA Key ......................... -
  VPN Client Policy Server Enabled ...... FALSE
  VPN Client Policy File Name ........... -
```

Table 45-21: Parameters in the output of the **show isakmp** command

Parameter	Meaning
Module Status	Whether ISAKMP is enabled or disabled.
UDP Port	The UDP port used for sending and receiving ISAKMP messages.
Debug Flag	A bit flag specifying debug options that are enabled: PKTRAW debugging (bit 0) STATE debugging (bit 1) TRACE debugging (bit 2) TRACEMORE debugging (bit 3) PACKET debugging (bit 4) A "0" means the option is disabled; a "1" means the option is enabled.
Global Debug Device	The device (port number) to which debugging output is sent.
Local RSA key	The ENCO key identification number of the RSA private key used for RSAENCR authentication.

Table 45-21: Parameters in the output of the **show isakmp** command

Parameter	Meaning
VPN Client Policy Server Enabled	Whether the router acts as a security policy server. If true, the router listens for security policy requests from remote ISAKMP peers. The router responds by sending the security policy specified by the **policyfilename** parameter. The **policyfilename** parameter is required and an ISAKMP policy for the peer must exist. The default is false. This feature is designed for the AT-VPN Client Windows software package. For more information on this parameter, see the AT-VPN Client documentation.
VPN Client Policy File Name	Specifies the security policy to be sent to remote ISAKMP peers when requested. This parameter must be specified when the **policyserverenabled** parameter is true or on. This feature is designed to use with the AT-VPN Client Windows software package. For more information on this parameter, see the AT-VPN Client documentation.

Examples To display the status of ISAKMP, use the command:

```
sh isa
```

Related Commands **disable isakmp**
enable isakmp

show isakmp counters

Syntax

```
SHow ISAkmp COUnters[={AGGressive|GENeral|HEArtbeat|INFo|
   IPSec|MAIn|NETwork|QUIck|SAD|SPD|TRAnsaction|XDB}]
```

Description This command displays all information counters for ISAKMP, or one or more categories of ISAKMP counters.

The **counters** parameter specifies the category or categories of counters to be displayed. If a counter category is not specified, all general ISAKMP counters are displayed. Multiple categories can be specified as a comma-separated list.

If **aggressive** is specified, counters for Aggressive Mode are displayed (Figure 45-26 on page 45-125, Table 45-22 on page 45-126).

If **general** is specified, general ISAKMP counters are displayed (Figure 45-27 on page 45-129, Table 45-23 on page 45-130).

If **heartbeat** is specified, heartbeat counters are displayed (Figure 45-27 on page 45-129,).

If **info** is specified, the counters for the informational exchanges are displayed (Figure 45-28 on page 45-134, Table 45-24 on page 45-134).

If **ipsec** is specified, the interface counters between ISAKMP and IPSEC are displayed (Figure 45-29 on page 45-136, Table 45-25 on page 45-136).

If **main** is specified, counters for the ISAKMP Main mode unit are displayed (Figure 45-30 on page 45-137, Table 45-26 on page 45-137).

If **network** is specified, counters for the transmission of ISAKMP messages over the network are displayed (Figure 45-31 on page 45-140, Table 45-27 on page 45-141).

If **quick** is specified, counters for the ISAKMP Quick mode are displayed (Figure 45-32 on page 45-142, Table 45-28 on page 45-143).

If **sad** is specified, counters for the ISAKMP Security Association Database (SAD) are displayed (Figure 45-33 on page 45-147, Table 45-29 on page 45-147).

If **spd** is specified, counters for the ISAKMP Security Policy Database (SPD) are displayed (Figure 45-34 on page 45-147, Table 45-30 on page 45-148).

If **transaction** is specified, counters for transaction exchanges are displayed (Figure 45-35 on page 45-148, Table 45-31 on page 45-149).

If **xdb** is specified, counters for the ISAKMP exchange database are displayed (Figure 45-36 on page 45-150, Table 45-32 on page 45-150).

Figure 45-26: Example output from the **show isakmp counter=aggressive** command

```
Aggressive Mode Counters:

 initSendSAKE                0   initSendSAKECallbackNoXchg  0
 initRecvSAKEAUTH            0   initRecvSAKEAUTHCbkNoXchg   0
 initSendAUTH                0   initSendAUTHCallbackNoXchg  0
 initSendNatD                0   initRecvNatD                0

 respRecvSAKE                0   respRecvSAKECallbackNoXchg  0
 respSendSAKEAUTH            0   respSendSAKEAUTHCbkNoXchg   0
 respRecvAUTH                0   respRecvAUTHCallbackNoXchg  0
 respSendNatD                0   respRecvNatD                0

 invalidSAKEMessage          0   invalidSAKEAUTHMessage      0
 invalidAUTHMessage          0   unexpectedMessage           0
 msgLengthIncorrect          0   reservedFieldNotZero        0
 generate3DESKeyFailed       0   generateDHPubFailed         0
 generateDHSecretFailed      0   generateKeysFailed          0
 generateKeysGood            0   generateLocalDHGood         0
 generateSharedSecretGood    0   generateSKEYIDaFailed       0
 generateSKEYIDdFailed       0   generateSKEYIDeFailed       0
 generateSKEYIDFailed        0

 decryptIDiiFailed           0   decryptIDiiGood             0
 decryptIDirFailed           0   decryptIDirGood             0
 decryptNiFailed             0   decryptNiGood               0
 decryptNrFaileD             0   decryptNrGood               0
 encryptIDiiFailed           0   encryptIDiiGood             0
 encryptIDirFailed           0   encryptIDirGood             0
 encryptNiFailed             0   encryptNiGood               0
 encryptNrFailed             0   encryptNrGood               0
 signFailed                  0   signGood                    0
 verifySigFailed             0

 invalidPolicy               0   noPolicy                    0
 noTransformChosen           0   saNotFirst                  0
 moreThanOneProposal         0   doiNotIpsec                 0
 situationNotSupported       0   proposalIdNotISAKMP         0
 proposalNumNotOne           0   tooManyTransforms           0
 transformInvalid            0   peerIdAndIpDifferent        0
 certNotSent                 0   certSent                    0
 authFailed                  0   authGood                    0
 cantFindLocalRSAKey         0   cantFindPeerRSAKey          0
 cantLoadSharedKey           0
```

Table 45-22: Parameters in the output of the **show isakmp counter=aggressive** command

Parameter	Meaning
initSendSAKE	The number of Security Association/Key Exchange messages sent.
initRecvSAKEAUTH	The number of Security Association/Key Exchange/ Authentication messages received.
initSendAUTH	The number of authentication messages sent.
initSendNatD	The number of NAT-D messages sent.
respRecvSAKE	The number of Security Association/Key Exchange messages received.
respSendSAKEAUTH	The number of Security Association/Key Exchange/ Authentication messages sent.
respRecvAUTH	The number of authentication messages received.
respSendNatD	The number of NAT-D messages sent.
initSendSAKECallbackNoXchg	The number of times that a Security Association/Key Exchange message was not sent due to an invalid reference from ENCO.
initRecvSAKEAUTHCbkNoXchg	The number of times that a Security Association/Key Exchange/Authentication message was not received due to an invalid reference from ENCO.
initSendAUTHCallbackNoXchg	The number of times that an authentication message was not sent due to an invalid reference from ENCO.
initRecvNatD	The number of NAT-D messages received.
respRecvSAKECallbackNoXchg	The number of times that a Security Association/Key Exchange message was not received due to an invalid reference from ENCO.
respSendSAKEAUTHCbkNoXchg	The number of times that a Security Association/Key Exchange/Authentication message was not sent due to an invalid reference from ENCO.
respRecvAUTHCallbackNoXchg	The number of times that an authentication message was not received due to an invalid reference from ENCO.
respRecvNatD	The number of NAT-D messages received.
invalidSAKEMessage	The number of invalid SA/Key Exchange messages received.
invalidAUTHMessage	The number of invalid authentication messages received.
msgLengthIncorrect	The number of messages received with an invalid message length.
generate3DESKeyFailed	The number of attempts to generate a 3DES key that failed.
generateDHSecretFailed	The number of attempts to generate the DH shared secret that failed.
generateKeysGood	The number of keys successfully generated.
generateSharedSecretGood	The number of times the DH shared secret was successfully generated.
generateSKEYIDdFailed	The number of attempts to generate SKEYIDd that failed.
generateSKEYIDFailed	The number of attempts to generate SKEYID that failed.
invalidSAKEAUTHMessage	The number of invalid SAKEAUTH messages received.

Table 45-22: Parameters in the output of the **show isakmp counter=aggressive** command (continued)

Parameter	Meaning
unexpectedMessage	The number of unexpected messages received.
reservedFieldNotZero	The number of messages received with a non-zero reserved field.
generateDHPubFailed	The number of attempts to generate the local DH secret that failed.
generateKeysFailed	The number of attempts to generate keys that failed.
generateLocalDHGood	The number of times the local DH secret was successfully generated.
generateSKEYIDaFailed	The number of attempts to generate SKEYIDa that failed.
generateSKEYIDeFailed	The number of attempts to generate SKEYIDe that failed.
decryptIDiiFailed	The number of attempts to decrypt the IDii payload that failed.
decryptIDirFailed	The number of attempts to decrypt the IDir payload that failed.
decryptNiFailed	The number of attempts to decrypt the Ni payload that failed.
decryptNrFailed	The number of attempts to decrypt the Nr payload that failed.
encryptIDiiFailed	The number of attempts to encrypt the IDii payload that failed.
encryptIDirFailed	The number of attempts to encrypt the IDir payload that failed.
encryptNiFailed	The number of attempts to encrypt the Ni payload that failed.
encryptNrFailed	The number of attempts to encrypt the Nr payload that failed.
signFailed	The number of attempts to sign the hash that failed.
verifySigFailed	The number of attempts to verify a signature that failed.
decryptIDiiGood	The number of times the IDii payload was successfully decrypted with RSA.
decryptIDirGood	The number of times the IDir payload was successfully decrypted with RSA.
decryptNiGood	The number of times the Ni payload was successfully decrypted with RSA.
decryptNrGood	The number of times the Nr payload was successfully decrypted with RSA.
encryptIDiiGood	The number of times the IDii payload was successfully encrypted with RSA.
encryptIDirGood	The number of times the IDir payload was successfully encrypted with RSA.
encryptNiGood	The number of times the Ni payload was successfully encrypted with RSA.
encryptNrGood	The number of times the Nr payload was successfully encrypted with RSA.
signGood	The number of times the hash was successfully signed.
invalidPolicy	The number of times an invalid policy was found.

Table 45-22: Parameters in the output of the **show isakmp counter=aggressive** command (continued)

Parameter	Meaning
noTransformChosen	The number of times a suitable policy could not be found.
moreThanOneProposal	The number of times more than one proposal was sent in an SA payload.
situationNotSupported	The number of SA payloads received with an unsupported situation.
proposalNumNotOne	The number of proposal payloads with ID not one.
transformInvalid	The number of invalid Transform payloads received.
certNotSent	The number of times a certificate could not be sent.
authFailed	The number of peer authentication attempts that failed.
cantFindLocalRSAKey	The number of attempts to load the local RSA key that failed.
cantLoadSharedKey	The number of attempts to load the shared key that failed.
noPolicy	The number of attempts to finds a suitable policy that failed.
saNotFirst	The number of times the SA payload was not the first payload in a received message.
doiNotIpsec	The number of times a received SA payload has an unsupported DOI.
proposalIdNotISAKMP	The number of times an invalid proposal ID has been received.
tooManyTransforms	The number of times that an invalid number of transforms have ben received.
peerIdAndIpDifferent	The number of times the source IP address differed from the peer ID.
certSent	The number of successful attempts to send a local certificate.
authGood	The number of successful peer authentications.
cantFindPeerRSAKey	The number of attempts to load the peer's RSA key that failed.

Figure 45-27: Example output from the **show isakmp counter=general** command

```
ISAKMP General Counters

  acquire                 0    acquireIsakmpDisabled  0
  acquireNoPolicy         0    acquireNoSa            0
  acquireEquivFound       0    acquirePhase1XcgFound  0
  acquireQueued           0
  acqPh1XcgCreateFailed   0    acqPh2XcgCreateFailed  0
  acquireStartPhase1      0    acquireStartPhase2     0

  acquirePrenegNoPolicy   0    acquirePrenegNoSa      0
  acqPrenegSaExists       0    acqPrenegExistingXcg   0
  acqPrenegStartPhase1    0    acqPrePh1XcgCreatFail  0

  msgTx                   0    msgTxd                 0
  txEncryptNoExchange     0    msgTxEncryptNoEncoPrc  0
  msgTxStartEncrypt       0
  txEncryptFail           0    txEncryptGood          0
  msgTxEncryptExpKBytes   0
  msgTxdDestroyMain       0    msgTxdDestroyQuick     0
  msgTxdQuickIpsecRslt    0

  txRetryFinishedMain     0    txRetryFinishedQuick   0
  txRetryTxd              0

  msgRx                   0    msgRxd                 0
  msgRxInconsistLengths   0    msgRxBadLength         0
  msgRxBadReserved        0    msgRxBadVersion        0
  msgRxPhase1NoPolicy     0    msgRxPhase2NoSa        0
  msgRxNoExchange         0
  msgRxNoExchangePhase1   0    msgRxPh1XcgCreateFail  0
  msgRxP1NoXcgNot1stMsg   0
  msgRxNoExchangePhase2   0    msgRxPh2XcgCreateFail  0
  msgRxEncrypted          0    msgRxEncryptdUnexpect  0
  msgRxDecryptNoEncoPrc   0    msgRxStartDecrypt      0
  msgRxPlain              0    msgRxPlainUnexpected   0
  msgRxBadPortIpChange    0    msgRxFailOldPort       0
  rxDecryptNoExchange     0    rxDecryptedBadLength   0
  rxDecryptGood           0    rxDecryptFail          0

  infoNoMatchingPolicy    0    infoPh1NotifyDisabled  0
  infoPh1NoDelAllowed     0    infoPh1ExchgStartFail  0
  infoPh2SANotFound       0    infoPh2SANotActive     0
  infoPh2NotifyDisabled   0    infoPh2DeleteDisabled  0
  infoPh2XchgStartFail    0

Heartbeat Mode Counters:
  startXchgInitiator      0    startXchgResponder     0
  initXchgComplete        0    respXchgComplete       0
  txMsg                   0    rxMsg                  0
  startXchgInitNotCfgd    0    startXchgRespNotCfgd   0
  rxUnexpectedPayload     0    rxInvalidSeqno         0
  rxHashUnexpectedLen     0    rxNotifyPayInvalid     0
  rxBadHash               0    rxExtraPayloads        0
Delete Delay Counters:
  deleteDelayStarted 18346    deleteDelayNotUsed     2
  deleteDelayPktsRxd      0    deleteDelayRetrySent   0
  deleteDelayConSent      0    deleteDelayConNoSA     0
  deleteDelayTxExceeded   0    deleteDelayPktDiff     0
```

Table 45-23: Parameters in the output of the **show isakmp counter=general** command

Parameter	Meaning
acquire	The number of IPsec requests to negotiate IPsec SAs.
acquireNoPolicy	The number of IPsec requests without a valid ISAKMP policy.
acquireEquivFound	The number of IPsec requests received while an equivalent acquire request was already in progress.
acquireQueued	The number of IPsec requests queued for processing.
acqPh1XcgCreateFailed	The number of failures to create a phase 1 exchange.
acquireStartPhase1	The number of phase 1 exchanges started for IPsec acquire requests.
acquireIsakmpDisabled	The number of IPsec acquire requests received while ISAKMP was disabled.
acquireNoSa	The number of IPsec acquire requests without a valid ISAKMP SA.
acquirePhase1XcgFound	The number of IPsec acquire requests received while a valid phase 1 exchange already existed.
acqPh2XcgCreateFailed	The number of failures to create phase 2 exchanges.
acquireStartPhase2	The number of phase 2 exchanges started.
acquirePrenegNoPolicy	The number of requests to prenegotiate a phase 1 SA without a valid policy.
acqPrenegSaExists	The number of requests to prenegotiate a phase 1 SA while a phase 1 SA already existed.
acqPrenegStartPhase1	The number of phase 1 exchanges started due to prenegotiate requests.
acquirePrenegNoSa	The number of prenegotiate requests without a valid phase 1 SA.
acqPrenegExistingXcg	The number of prenegotiate requests received while a valid exchange already existed.
acqPrePh1XcgCreatFail	The number of prenegotiate requests where a phase 1 exchange could not be created.
msgTx	The number of message transmissions started by ISAKMP.
txEncryptNoExchange	The number of messages encrypted without a valid exchange.
msgTxStartEncrypt	The number of messages passed to the encryption process.
txEncryptFail	The number of messages that failed the encryption process.
msgTxEncryptExpKBytes	The number of encrypted messages that caused the SA to expire.
msgTxdDestroyMain	The number of messages transmitted that caused a Main mode exchange to be completed.
msgTxdQuickIpsecRslt	The number of messages transmitted where IPsec was informed of a successful negotiation.
msgTxd	The number of message transmissions completed by ISAKMP.
msgTxEncryptNoEncoPrc	The number of messages not encrypted because ENCO resources were not available.
txEncryptGood	The number of messages to be transmitted that were successfully encrypted.

Table 45-23: Parameters in the output of the **show isakmp counter=general** command

Parameter	Meaning
msgTxdDestroyQuick	The number of messages transmitted that caused a Quick Mode exchange to be completed.
txRetryFinishedMain	The number of retransmits completed for Main mode exchanges.
txRetryTxd	The number of retransmits of ISAKMP messages.
txRetryFinishedQuick	The number of retransmits completed for Quick Mode exchanges.
msgRx	The number of ISAKMP messages received.
msgRxInconsistLengths	The number of ISAKMP messages received with an invalid packet length.
msgRxBadReserved	The number of ISAKMP messages received in which the reserved field was not set to zero. This may indicate the decryption failed due to an invalid pre-shared key.
msgRxPhase1NoPolicy	The number of Main mode messages received with no valid ISAKMP policy.
msgRxNoExchange	The number of messages received with no valid exchange.
msgRxNoExchangePhase1	The number of messages received with no valid phase 1 exchange.
msgRxP1NoXcgNot1stMsg	The number of messages received with no exchange that are not the first message of an exchange.
msgRxNoExchangePhase2	The number of messages received with no valid phase 2 exchange.
msgRxEncrypted	The number of encrypted messages received.
msgRxDecryptNoEncoPrc	The number of encrypted messages received with no valid ENCO resources.
msgRxPlain	The number of messages received in plaintext.
msgRxBadPortIpChange	The number of ISAKMP packets received with an unexpected source IP address or source port and discarded.
rxDecryptNoExchange	The number of ISAKMP messages decrypted by ENCO without a valid existing exchange.
rxDecryptGood	The number of ISAKMP messages decrypted by ENCO.
msgRxd	The number of messages received by ISAKMP.
msgRxBadLength	The number of messages received with a bad length.
msgRxBadVersion	The number of messages received with a bad version number.
msgRxPhase2NoSa	The number of Quick Mode messages received without a valid ISAKMP SA.
msgRxPh1XcgCreateFail	The number of messages received where a phase 1 exchange could not be created.
msgRxPh2XcgCreateFail	The number of messages received where a phase 2 exchange could not be created.
msgRxEncryptdUnexpect	The number of encrypted messages received that should not have been encrypted.
msgRxStartDecrypt	The number of messages received that started the decryption process.

Table 45-23: Parameters in the output of the **show isakmp counter=general** command

Parameter	Meaning
msgRxPlainUnexpected	The number of messages received in plaintext that should have been encrypted.
msgRxFailOldPort	The number of ISAKMP packets discarded because they were received on port 500 after NAT-T had moved ISAKMP traffic to port 4500.
rxDecryptedBadLength	The number of ISAKMP messages decrypted with a bad length.
rxDecryptFail	The number of encrypted ISAKMP messages that were not successfully decrypted.
infoNoMatchingPolicy	The number of Info messages received without a matching policy.
infoPh1NoDelAllowed	The number of delete payloads received not in phase 2.
infoPh2SANotFound	The number of phase 2 messages received without a valid SA.
infoPh2NotifyDisabled	The number of Notify payloads received/sent when notifies are disabled in phase 2.
infoPh2XchgStartFail	The number of failed attempts to start an Info phase 2 exchange.
infoPh1NotifyDisabled	The number of Notify payloads received/sent when notifies are disabled in phase 2.
infoPh1ExchgStartFail	The number of failed attempts to start an Info phase 1 exchange.
infoPh2SANotActive	The number of failed attempts to start an Info exchange when the SA is not active.
infoPh2DeleteDisabled	The number of Delete payloads received/sent when deletes are disabled in phase 2.
Heartbeat Mode counters	**Counters about ISAKMP heartbeat mode exchanges**
startXchgInitiator	The number of times a heartbeat exchange started as the initiator
startXchResponder	The number of times a heartbeat exchange started as the responder.
initXchgComplete	The number of times a heartbeat exchange successfully started as the initiator.
respXchgComplete	The number of times a heartbeat exchange successfully started as the responder.
txMsg	The number of heartbeat messages sent.
rxMsg	The number of heartbeat messages received.
rxUnexpectedPayload	The number of heartbeat messages received with an unexpected payload.
startXchgInitNotCfgd	The number of times a heartbeat exchange attempted to start when the local device was not configured to initiate heartbeats.
startXchgRespNotCfgd	The number of times a heartbeat exchange was initiated by a remote device when the local device was not configured to receive heartbeats.
rxInvalidSeqno	The number of heartbeat messages received with an invalid sequence number.

Table 45-23: Parameters in the output of the **show isakmp counter=general** command

Parameter	Meaning
rxHashUnexpectedLen	The number of heartbeat messages received with a hash payload with an unexpected length.
rxNotifyPayInvalid	The number of heartbeat messages received with an invalid notify payload.
rxBadHash	The number of heartbeat messages received with a bad hash value.
rxExtraPayloads	The number of heartbeat messages received with unexpected extra payloads.
Delete Delay Counters	**Counters about ISAMP Delete Delay**
deleteDelayStarted	The number of ISAKMP exchanges where a deletedelay period was started.
deleteDelayNotUsed	The number of ISAKMP exchanges where a deletedelay period was not used. This may occur if the deletedelay is zero, if the device did not send the last message in the exchange, or for informational exchanges, where the peer may not send a message in the exchange.
deleteDelayPktsRxd	The number of retransmissions received from ISAKMP peers during the deletedelay periods.
deleteDelayRetrySent	The number of retransmitted messages the device has sent during the deletedelay periods.
deleteDelayConSent	The number of retransmitted informational connected messages the device has sent during the deletedelay periods.
deleteDelayConNoSa	The number of informational connected messages the device failed to retransmit due to lack of a suitable ISAKMP SA.
deleteDelayTxExceeded	The number of deletedelay periods where the retries sent exceeds the ISAKMP policy's msgretrylimit.
deleteDelayPktDiff	The number of retransmissions received that have a different unencrypted length or next payload type compared to the last received packet.

Figure 45-28: Example output from the **show isakmp counter=info** command

```
Informational Message Counters:

  startXchgInitiator              0  startXchgResponder              0
  startInfoXchgProtected          0  startInfoXchgUnprotected        0
  rxMsgWithHashNoCryptInfo        0  rxMsgHashFail                   0
  rxMsgHashSuccess                0  rxMsgUnexpectHashPayload        0
  rxMsgProcessNotifyPayload       0  rxMsgProcessDeletePayload       0
  rxMsgProcessNotifyPayloadErr    0  rxMsgProcessDeletePayloadErr    0
  rxMsgUnexpectedPayload          0  rxMsgDelPayloadUnprotect        0
  rxMsgNonZeroReservedField       0  rxMsgProcessingComplete         0
  txMsgNotifyDenyIkmpPolicy       0  txMsgDeleteDenyIkmpPolicy       0
  txMsgSentCallBack               0  txMsgCreateXchgFailed           0
  txMsgStartPhase1XchgFail        0  txMsgStartPhase2XchgFail        0
  txNotifyMsg                     0  txDeleteMsg                     0
  txMsgDeleteSANotFound           0  txMsgDeleteDOINotValid          0
  processNotify                   0  processDelete                   0
  processNotifyResNotZero         0  processNotifyDoiInvalid         0
  processNotifyInvalidProtId      0  processDeleteReservedNotZero    0
  processDeleteDoiInvalid         0  processDeleteInvalidProtId      0
  createNotifyDoiInvalid          0  createNotifyMsgTypeReserved     0
  createNotifyMsgTypePrivate      0  xchgCompleteSucessfull          0
```

.
Table 45-24: Parameters in the output of the **show isakmp counter=info** command

Parameter	Meaning
startXchgInitiator	The number of information exchanges started as initiator.
startXchgResponder	The number of information exchanges started as responder.
startInfoXchgProtected	The number of Information exchanges started with SA protection.
startInfoXchgUnprotected	The number of Information exchanges started with no protection.
rxMsgWithHashNoCryptInfo	The number of Rx protected Information messages unable to validate the hash.
rxMsgHashFail	The number of Rx protected information messages with invalid hash information.
rxMsgHashSuccess	The number Rx protected information messages with valid hash information.
rxMsgUnexpectHashPayload	The number of Rx information messages with hash payload unexpected.
rxMsgProcessNotifyPayload	The number of Notify payloads processed successfully.
rxMsgProcessDeletePayload	The number of Delete payloads processed successfully.
rxMsgProcessNotifyPayloadErr	The number of Notify payloads processed with errors.
rxMsgProcessDeletePayloadErr	The number of Delete payloads processed with errors.
rxMsgUnexpectedPayload	The number of Information messages received with unexpected payloads.
rxMsgDelPayloadUnprotect	The number of Delete payloads received with no protection.
rxMsgNonZeroReservedField	The number of Information messages received with a non-zero reserved field.
rxMsgProcessingComplete	The number of Informational messages processed.

Table 45-24: Parameters in the output of the **show isakmp counter=info** command

Parameter	Meaning
txMsgNotifyDenyIkmpPolicy	The number of attempts to send Notify messages denied by the ISAKMP policy.
txMsgDeleteDenyIkmpPolicy	The number of attempts to send Delete messages denied by the ISAKMP policy.
txMsgSentCallBack	The number of Information messages transmitted.
txMsgCreateXchgFailed	The number of attempts to send Information messages that failed to create an exchange.
txMsgStartPhase1XchgFail	The number of failed attempts to start phase1 informational exchange.
txMsgStartPhase2XchgFail	The number of failed attempts to start phase2 informational exchange.
txNotifyMsg	The number of Notify messages transmitted.
txDeleteMsg	The number of Delete messages transmitted.
txmsgdeletesanotfound	The number of attempts to transmit Delete messages where no phase 1 SA was found.
txMsgDeleteDOINotValid	The number of attempts to transmit Delete messages with an invalid DOI.
processNotify	The number of Notify payloads processed.
processDelete	The number of Delete payloads processed.
processNotifyResNotZero	The number of Notify payloads processed with the reserved field not zero.
processNotifyDoiInvalid	The number of Notify payloads processed with an invalid DOI.
processNotifyInvalidProtId	The number of Notify payloads processed with the protocol identity number is not valid.
processDeleteReservedNotZero	The number of Delete payloads processed with the reserved field not zero.
processDeleteDoiInvalid	The number of Delete payloads processed with an invalid DOI.
processDeleteInvalidProtId	The number of Delete payloads processed with an invalid protocol identity number.
createNotifyDoiInvalid	The number of Notify payloads created with an invalid DOI.
createNotifyMsgTypeReserved	The number of attempts to create Notify payloads with Notify type in the reserved range.
createNotifyMsgTypePrivate	The number of attempts to create Notify payloads with Notify type in the private range.
xchgCompleteSucessful	The number of Information exchanges completed.

Figure 45-29: Example output from the **show isakmp counter=ipsec** command

```
IPSEC DOI Counters:

 acquire                    2   acquireFailedDisabled      0
 commit                     0   commitFailedDisabled       0
 deleteSaBadLength          0
 checkMatchAttrBadType      0   checkMatchAttrSeenTwice    0
 chkMtchAttrsBadLen         0   chkMtchAttrsLifValMsng     0
 chkMtchAttrLifValBadLen    0   chkMtchAttrsBadVal         0
 chkMtchAttrsLifTypeMsng    0   chkMtchAttrssUnsupp        0
 chkMtchAttrsVarUnexp       0
```

Table 45-25: Parameters in the output of the **show isakmp counter=ipsec** command

Parameter	Meaning
acquire	The number of IPsec requests to negotiate IPsec SAs.
commit	The number of committed IPsec bundles.
deleteSaBadLength	The number of attempted SA deletions where the SPI length is invalid.
checkMatchAttrBadType	The number of bad attribute types received.
chkMtchAttrsBadLen	The number of attributes received with bad lengths.
chkMtchAttrLifValBadLen	The number of life duration attributes received where the attribute length is invalid.
chkMtchAttrsLifTypeMsng	The number of life duration attributes received without a matching life type attribute.
chkMtchAttrsVarUnexp	The number of unexpected attributes received.
acquireFailedDisabled	The number of times IPsec requested an SA negotiation and ISAKMP is disabled.
commitFailedDisabled	The number of IPsec bundles committed when ISAKMP is disabled.
checkMatchAttrSeenTwice	The number of attributes that have been seen twice.
chkMtchAttrsLifValMsng	The number of life type attributes received without a matching life duration attribute.
chkMtchAttrsBadVal	The number of attributes received with invalid values.
chkMtchAttrssUnsupp	The number of unsupported attributes received.

Figure 45-30: Example output from the **show isakmp counter=main** command

```
Main Mode Counters:

initSendSA                    0   initRecvSA                    0
initSendKE                    0   initSendKECallbackNoXchg      0
initSendNatD                  0   initRecvNatD                  0
initRecvKE                    0   initRecvKECallbackNoXchg      0
initSendAUTH                  0   initSendAUTHCallbackNoXchg    0
initRecvAUTH                  0   initRecvAUTHCallbackNoXchg    0

respRecvSA                    0  respRecvSACallbackNoXchg       0
respSendSA                    0  respSendKE                     0
respSendNatD                  0  respRecvNatD                   0
respSendKECallbackNoXchg      0  respRecvKE                     0
respRecvKECallbackNoXchg      0  respRecvAUTH                   0
respRecvAUTHCallbackNoXchg    0  respSendAUTH                   0
respSendAUTHCallbackNoXchg    0

invalidSAMessage              0   invalidKEMessage              0
invalidAUTHMessage            0   unexpectedMessage             0
msgLengthIncorrect            0   reservedFieldNotZero          0
generate3DESKeyFailed         0   generateDHPubFailed           0
generateDHSecretFailed        0   generateKeysFailed            0
generateKeysGood              0   generateLocalDHGood           0
generateSharedSecretGood      0   generateSKEYIDaFailed         0
generateSKEYIDdFailed         0   generateSKEYIDeFailed         0
generateSKEYIDFailed          0

decryptIDiiFailed             0   decryptIDiiGood               0
decryptIDirFailed             0   decryptIDirGood               0
decryptNiFailed               0   decryptNiGood                 0
decryptNrFailed               0   decryptNrGood                 0
encryptIDiiFailed             0   encryptIDiiGood               0
encryptIDirFailed             0   encryptIDirGood               0
encryptNiFailed               0   encryptNiGood                 0
encryptNrFailed               0   encryptNrGood                 0
signFailed                    0   signGood                      0
verifySigFailed               0

invalidPolicy                 0   noPolicy                      0
noTransformChosen             0   saNotFirst                    0
moreThanOneProposal           0   doiNotIpsec                   0
situationNotSupported         0   proposalIdNotISAKMP           0
proposalNumNotOne             0   tooManyTransforms             0
transformInvalid              0   peerIdAndIpDifferent          0
certNotSent                   0   certSent                      0
authFailed                    0   authGood                      0
cantFindLocalRSAKey           0   cantFindPeerRSAKey            0
cantLoadSharedKey             0
```

Table 45-26: Parameters in the output of the **show isakmp counter=main** command

Parameter	Meaning
Initiator Counters	
initSendSA	The number of Security Association messages sent.
initSendKE	The number of key exchange messages sent.
initSendNatD	The number of NAT-D messages sent.
initRecvKE	The number of key exchange messages received.

Table 45-26: Parameters in the output of the **show isakmp counter=main** command

Parameter	Meaning
initSendAUTH	The number of authentication messages sent.
initRecvAUTH	The number of authentication messages received.
initRecvSA	The number of Security Association messages received.
initSendKECallbackNoXchg	The number of times that a key exchange message was not sent due to an invalid reference from ENCO.
initRecvNatD	The number of NAT-D messages received.
initRecvKECallbackNoXchg	The number of times that a key exchange message was not received due to an invalid reference from ENCO.
initSendAUTHCallbackNoXchg	The number of times that an authentication payload was not sent due to an invalid reference from ENCO.
initRecvAUTHCallbackNoXchg	The number of times that an authentication payload was not received due to an invalid reference from ENCO.
Responder Counters	
respRecvSA	The number of Security Association messages received.
respSendSA	The number of Security Association messages sent.
respSendNatD	The number of NAT-D messages sent.
respSendKECallbackNoXchg	The number of times that a key exchange was not sent due to an invalid reference from ENCO.
respRecvKECallbackNoXchg	The number of times that a key exchange message was not received due to an invalid reference from ENCO.
respSendAUTHCallbackNoXchg	The number of times that an authentication payload was not sent due to an invalid reference from ENCO.
respRecvAUTHCallbackNoXchg	The number of times that an authentication payload was not received due to an invalid reference from ENCO.
respRecvSACallbackNoXchg	The number of times that an SA payload was not received due to an invalid reference from ENCO.
respSendKE	The number of key exchange messages sent.
respRecvNatD	The number of NAT-D messages received.
respRecvKE	The number of key exchange messages received.
respRecvAUTH	The number of authentication messages received.
respSendAUTH	The number of authentication messages sent.
invalidSAMessage	The number of invalid SA messages received.
invalidAUTHMessage	The number of invalid authentication messages received.
msgLengthIncorrect	The number of messages received with an invalid message length.
generate3DESKeyFailed	The number of failed attempts to generate a 3DES key.
generateDHSecretFailed	The number of attempts to generate the DH shared secret that failed.
generateKeysGood	The number of keys successfully generated.
generateSharedSecretGood	The number of times the DH shared secret was successfully generated.
generateSKEYIDdFailed	The number of failed attempts to generate SKEYIDd.
generateSKEYIDFailed	The number of failed attempts to generate SKEYID.
invalidKEMessage	The number of invalid KE messages received.

Table 45-26: Parameters in the output of the **show isakmp counter=main** command

Parameter	Meaning
unexpectedMessage	The number of unexpected messages received.
reservedFieldNotZero	The number of messages received with a non-zero reserved field.
generateDHPubFailed	The number of attempts to generate the local DH secret that failed.
generateKeysFailed	The number of attempts to generate keys that failed.
generateLocalDHGood	The number of times the local DH secret was successfully generated.
generateSKEYIDaFailed	The number of attempts to generate SKEYIDa that failed.
generateSKEYIDeFailed	The number of attempts to generate SKEYIDe that failed.
decryptIDiiFailed	The number of attempts to decrypt the IDii payload that failed.
decryptIDirFailed	The number of attempts to decrypt the IDir payload that failed.
decryptNiFailed	The number of attempts to decrypt the Ni payload that failed.
decryptNrFailed	The number of attempts to decrypt the Nr payload that failed.
encryptIDiiFailed	The number of attempts to encrypt the IDii payload that failed.
encryptIDirFailed	The number of attempts to encrypt the IDir payload that failed.
encryptNiFailed	The number of attempts to encrypt the Ni payload that failed.
encryptNrFailed	The number of attempts to encrypt the Nr payload that failed.
signFailed	The number of attempts to sign the hash that failed.
verifySigFailed	The number of attempts to verify a signature that failed.
decryptIDiiGood	The number of times the IDii payload was successfully decrypted with RSA.
decryptIDirGood	The number of times the IDir payload was successfully decrypted with RSA.
decryptNiGood	The number of times the Ni payload was successfully decrypted with RSA.
decryptNrGood	The number of times the Nr payload was successfully decrypted with RSA.
encryptIDiiGood	The number of times the IDii payload was successfully encrypted with RSA.
encryptIDirGood	The number of times the IDir payload was successfully encrypted with RSA.
encryptNiGood	The number of times the Ni payload was successfully encrypted with RSA.
encryptNrGood	The number of times the Nr payload was successfully encrypted with RSA.
signGood	The number of times the hash was successfully signed.
invalidPolicy	The number of times an invalid policy was found.

Table 45-26: Parameters in the output of the **show isakmp counter=main** command

Parameter	Meaning
noTransformChosen	The number of times a suitable policy could not be found.
moreThanOneProposal	The number of times more than one proposal was sent in an SA payload.
situationNotSupported	The number of times a received SA payload had an unsupported situation.
proposalNumNotOne	The number of times the proposal payload did not have an ID of one.
transformInvalid	The number of invalid transform payloads received.
certNotSent	The number of times a certificate could not be sent.
authFailed	The number of peer authentication attempts that failed.
cantFindLocalRSAKey	The number of attempts to load the local RSA key that failed.
cantLoadSharedKey	The number of attempts to load the shared key that failed.
noPolicy	The number of attempts to finds a suitable policy that failed.
saNotFirst	The number of times the SA payload was not the first payload in a received message.
doiNotIpsec	The number of times a received SA payload had an unsupported DOI.
proposalIdNotISAKMP	The number of times an invalid proposal ID was received.
tooManyTransforms	The number of times an invalid number of transforms was received.
peerIdAndIpDifferent	The number of times the source IP address differed from the peer ID.
certSent	The number of successful attempts to send a local certificate.
authGood	The number of successful peer authentications.
cantFindPeerRSAKey	The number of failed attempts to load the peer's RSA key.

Figure 45-31: Example output from the **show isakmp counter=network** command

```
ISAKMP Network Counters

  opened                  0     openFail                0
  openAlreadyOpened       0
  closed                  0     closeFail               0
  closeNotOpened          0
  rx                      0     rxFailNoNonEspMarker    0
  tx                      0     txFail                  0
  txNotOpened             0
```

Table 45-27: Parameters in the output of the **show isakmp counter=network** command

Parameter	Meaning
opened	The number of successful attempts to open the network interface.
openAlreadyOpened	The number of attempts to open the network interface when it was already open.
closed	The number of successful attempts to close the network interface.
closeNotOpened	The number of attempts to close the network interface when it was already closed.
rx	The number of ISAKMP messages received.
tx	The number of ISAKMP messages transmitted.
txNotOpened	The number of attempts to transmit an ISAKMP message when the network interface was not open.
openFail	The number of failed attempts to open the network interface.
closeFail	The number of failed attempts to close the network interface.
rxFailNoNonEspMarker	The number of packets ISAKMP dropped because they were received on NAT-T port 4500 without a non-ESP marker.
txFail	The number of failed attempts to transmit an ISAKMP message.

Figure 45-32: Example output from the **show isakmp counter=quick** command

```
Quick Mode Counters:

General Counters:
        Initiator:
    startExchange         0     exchangeGood            0
    hashSaNonceSent       0     hashSent                0
    hashSaNonceReceived   0     hashSaNonceRcvdGood     0
    connectedReceived     1     connectedReceivedGood   1
    natOaSent             0     natOaReceived           0
    hashSaNonceExpKByte   0     connectedExpKByte       0

        Responder:
    startExchange         0     exchangeGood            0
    hashSaNonceSent       0
    hashSaNonceReceived   0     hashSaNonceRcvdGood     0
    hashReceived          0     hashReceivedGood        0
    hashSaNonceExpKByte   0     hashExpKByte            0
    natOaSent             0     natOaReceived           0

 Error Counters:

        Initiator: General errors:
    initHash2Fail         0     initDHGenFail           0
    initStartCBFailed     0     initStartCBNoXchg       0
    initProc1CBFailed     0     initProc1CBNoXchg       0
    initHash4Fail         1

                   Receive Hash SA Nonce message errors:
    invalidPayloadType    0     hashPayMissing          0
    hashUnexpectedLen     0
    saNoMatch             0     saBadLen                0
    saBadDoi              0     saBadSituation          0
    saLenTooShort         0     saPayMissing            0
    saPropInconsistLen    0
    propNoMatch           0     propBadLen              0
    propTranInconsistLen  0     propBadNextPayload      0
    propBadRsv            0     propMultipleTrans       0
    propTooManyProps      0
    tranBadLen            0     tranBadNextPayload      0
    tranBadRsv            0     attrBad                 0
    nonceBadLen           0     noncePayMissing         0
    nonceSeenTwice        0     idsSeenTwice            0
    idciBad               0     idcrBad                 0
    idcrPayMissing        0     keSeenTwice             0
    badNatOa              0

                   Receive Connected message errors:
    hashPayMissing        0     invalidPayloadType      0
    hashBadLen            0

        Responder: General errors:
    respAcquireNoPolicy   0     respRemotePropNoMatch   0
    respHash1Fail         0     respHash3Fail           0
    respProc2CBFailed     0     respProc2CBNoXchg       0
    respProc3CBFailed     0     respProc3CBNoXchg       0
```

Figure 45-32: Example output from the **show isakmp counter=quick** command (continued)

```
                  Receive Hash SA Nonce message errors:
    invalidPayloadType   0     hashPayMissing          0
    hashUnexpectedLen    0     noncePayMissing         0
    nonceSeenTwice       0     nonceBadLen             0
    saPayMissing         0     saPayBad                0
    saLenTooLong         0     saLenTooShort           0
    saBadDoi             0     saBadSituation          0
    saPropInconsistLen   0
    propLenTooLong       0     propLenTooShort         0
    propBadLen           0     propBadRsv              0
    propBadNextPayload   0     propBadNumber           0
    propBadProtid        0     propBadSpiSize          0
    propNoTransforms     0     propTranInconsistLen    0
    tranBadLen           0     tranBadNextPayload      0
    tranBadNumber        0     tranBadRsv2             0
    tranTransId          0     attrBad                 0
    idsSeenTwice         0
    idciBadType          0     idcrBadType             0
    idcrPayMissing       0     keSeenTwice             0
    badNatOa             0

              Receive Hash message errors:

    hashPayMissing       0     invalidPayloadType      0
    hashBadLen           0
```

Table 45-28: Parameters in the output of the **show isakmp counter=quick** command

Parameter	Meaning
General counters for this router as initiator or responder	
startExchange	The number of times an exchange was started.
exchangeGood	The number of times an exchange was successfully completed.
hashSaNonceSent	The number of times a hash SA nonce message was sent.
hashSent	The number of times a hash message was sent.
hashSaNonceReceived	The number of times a hash SA nonce message was received.
hashSaNonceRcvdGood	The number of times a valid hash SA nonce message was received and processed successfully.
connectedReceived	The number of times a Connected message was received in Quick mode.
connectedReceivedGood	The number of times a valid Connected message was received in Quick mode and processed successfully.
natOaSent	The number of NAT originating addresses sent.
natOaReceived	The number of NAT originating addresses received.
hashSaNonceExpKByte	The number of times a hash SA nonce message was received and caused the exchange to expire due to reaching its Kbyte limit.
connectedExpKByte	The number times a Connected message was received in Quick mode and caused the SA to expire due to reaching its Kbyte limit.
hashReceived	The number of times a hash message was received.

Table 45-28: Parameters in the output of the **show isakmp counter=quick** command

Parameter	Meaning
hashReceivedGood	The number of times a valid hash message was received and processed successfully.
hashExpKByte	The number of times a hash message was received and caused the exchange to expire due to reaching its Kbyte limit.
Error counters for initiators and responders	
initHash2Fail	The number of times the second hash check of an exchange failed.
initDHGenFail	The number of times the Diffie-Hellman generation for an exchange failed.
initStartCBFailed	The number of times a call to the ENCO module failed during the start of an exchange initiaited by this router.
initStartCBNoXchg	The number of times the specified exchange could not be found after returning from a call to the ENCO module during the start of an exchange initiaited by this router.
initProc1CBFailed	The number of times a call to the ENCO module failed during the processing of the first message of an exchange.
initProc1CBNoXchg	The number of times the specified exchange could not be found after returning from a call to the ENCO module during the processing of the first message of an exchange.
respAcquireNoPolicy	The number of times the first message of an exchange was received and an IPsec policy to match the policy proposed in the message could not be found.
respRemotePropNoMatch	The number of times the first message of an exchange was received and a proposal in the message did not match any proposal in the selected IPsec policy.
respHash1Fail	The number of times the first hash check of an exchange failed.
respHash3Fail	The number of times the third hash check of an exchange failed.
respProc2CBFailed	The number of times a call to the ENCO module failed during the processing of the second message of an exchange.
respProc2CBNoXchg	The number of times the specified exchange could not be found after returning from a call to the ENCO module during the processing of the second message of an exchange.
respProc3CBFailed	The number of times a call to the ENCO module failed during the processing of the third message of an exchange.
respProc3CBNoXchg	The number of times the specified exchange could not be found after returning from a call to the ENCO module during the processing of the third message of an exchange.
Receive hash SA nonce message errors for initiators and responders	
invalidPayloadType	The number of times a message was received with an invalid payload type.
hashPayMissing	The number of times a message was received with a hash payload missing.

Table 45-28: Parameters in the output of the **show isakmp counter=quick** command

Parameter	Meaning
hashUnexpectedLen	The number of times a message was received with a hash payload with an unexpected length.
saNoMatch	The number of times a message was received with an SA payload that did not match any local ISAKMP policy.
saBadLen	The number of times a message was received with an SA payload with a bad length.
saBadDoi	The number of times a message was received with an SA payload with an invalid DOI field.
saBadSituation	The number of times a message was received with an SA payload with an invalid situation field.
saLenTooShort	The number of times a message was received with an SA payload with a length that was too short.
saPayMissing	The number of times a message was received with no SA payload when one was expected.
saPropInconsistLen	The number of times a message was received with an SA payload with a length that was inconsistent with the lengths of the proposal payloads contained in the SA payload.
propNoMatch	The number of times a proposal payload was received that did not match a proposal in the local ISAKMP policy.
propBadLen	The number of times a message was received with a proposal payload with a bad length.
propTranInconsistLen	The number of times a message was received with a proposal payload with a length that was inconsistent with the lengths of the transform payloads contained in the proposal payload.
propBadNextPayload	The number of times a message was received with a proposal payload with a bad next payload field.
propBadRsv	The number of times a message was received with a proposal payload with a bad reserved field.
propMultipleTrans	The number of times a message was received by the initiator with a proposal payload with multiple transforms.
propTooManyProps	The number of times a message was received by the initiator with a proposal payload with too many proposals.
tranBadLen	The number of times a message was received with a transform payload with a bad length.
tranBadNextPayload	The number of times a message was received with a transform payload with a bad next payload field.
tranBadRsv	The number of times a message was received with a transform payload with a bad reserved field.
attrBad	The number of times a message was received with a bad attribute.
nonceBadLen	The number of times a message was received with a nonce payload with a bad length.
noncePayMissing	The number of times a message was received without a nonce payload when one was expected
nonceSeenTwice	The number of times a message was received with more than one nonce payload.

Table 45-28: Parameters in the output of the **show isakmp counter=quick** command

Parameter	Meaning
idsSeenTwice	The number of times a message was received with more than one pair of ID payloads.
idciBad	The number of times a message was received with a bad IDci payload.
idcrBad	The number of times a message was received with a bad IDcr payload.
idcrPayMissing	The number of times a message was received with an IDci payload and no matching IDcr payload.
keSeenTwice	The number of times a message was received with more than one key exchange payload.
badNatOa	The number of non-conforming NAT-OA (originating address) messages received such as an unknown type.
saPayBad	The number of times a message was received with a bad SA payload.
saLenTooLong	The number of times a message was received with an SA payload with a length that was too long.
propLenTooLong	The number of times a message was received with a proposal payload with a length that was too long.
propLenTooShort	The number of times a message was received with a proposal payload with a length that was too short.
propBadNumber	The number of times a message was received with a proposal payload with a bad proposal number.
propBadProtid	The number of times a message was received with a proposal payload with a bad protocol Id field.
propBadSpiSize	The number of times a message was received with a proposal payload with a bad SPI size field.
propNoTransforms	The number of times a message was received with a proposal payload with no transforms.
tranBadNumber	The number of times a message was received with a transform payload with a bad transform number field.
tranBadRsv2	The number of times a message was received with a transform payload with a bad reserved2 field.
tranTransId	The number of times a message was received with a transform payload with a bad transform ID field.
idciBadType	The number of times a message was received with a IDci payload with a bad ID type field.
idcrBadType	The number of times a message was received with a IDcr payload with a bad ID type field.
(Responder) **Receive hash message errors**	
hashPayMissing	The number of times a message was received with a hash payload missing.
invalidPayloadType	The number of times a message was received with an invalid payload type.
hashBadLen	The number of times a message was received with a hash payload with a bad length.

Figure 45-33: Example output from the **show isakmp counter=sad** command

```
SAD Counters
  createGood           0
  deleteGood           0      deleteFailed         0
  expireNoSa           0      softExpireNoSa       0
  Find SA Counters:    success               fail
    by Id                   0                   0
    by peer address         0                   0
    by cookies              0                   0
    by policy name          0                   0
    by peer Notify          0                   0
    by peer Delete          0                   0
```

Table 45-29: Parameters in the output of the **show isakmp counter=sad** command

Parameter	Meaning
createGood	The number of Security Associations created successfully.
deleteGood	The number of Security Associations deleted successfully.
expireNoSa	The number of times the matching SA could not be found during an expiry event.
deleteFailed	The number of delete Security Association operations that failed.
softExpireNoSa	The number of times the matching SA could not be fould during a soft expiry event.
Find SA Counters	Counters for Security Association search operations.
success	The number of successful searches of a given type.
fail	The number of unsuccessful searches of a given type.
by Id	The number of searches for a Security Association with a specified ID.
by peer address	The number of searches for a Security Association with a specified peer address.
by cookies	The number of searches for a Security Association with a specified cookies.
by policy name	The number of searches for a Security Association with a specified IPsec policy name.
by peer Notify	The number of searches for an active Security Association suitable for protecting an informational notify exchange to a specified peer address.
by peer Delete	The number of searches for an active Security Association suitable for protecting an informational delete exchange to a specified peer address.

Figure 45-34: Example output from the **show isakmp counter=spd** command

```
ISAKMP Policy Counters

  getPolicyGood        0    getPolicyFailed        1
  deletePolicyGood     0    deletePolicyFailed     0
  addPolicyGood        0    addPolicyFailed        0
  getPolicyByPeerGood  0    getPolicyByPeerFailed  0
```

Table 45-30: Parameters in the output of the **show isakmp counter=spd** command

Parameter	Meaning
getPolicyGood	The number of successful attempts to retrieve a policy from the ISAKMP SPD.
deletePolicyGood	The number of successful attempts to delete a policy from the ISAKMP SPD.
addPolicyGood	The number of successful attempts to add a policy to the ISAKMP SPD.
getPolicyByPeerGood	The number of successful attempts to retrieve a policy using the peer IP address.
getPolicyFailed	The number of unsuccessful attempts to retrieve a policy from the ISAKMP SPD.
deletePolicyFailed	The number of unsuccessful attempts to delete a policy from the ISAKMP SPD.
addPolicyFailed	The number of unsuccessful attempts to add a policy to the ISAKMP SPD.
getPolicyByPeerFailed	The number of unsuccessful attempts to retrieve a policy using the peer IP address.

Figure 45-35: Example output from the **show isakmp counter=transaction** command

```
Transaction Exchange Counters:

  msgLengthIncorrect         0   reservedFieldNotZero       0
  tooManyAttributes          0   unexpectedAttributes       0
  hashNotFirst               0   hashTooBig                 0
  hashInvalid                0   unexpectedMessage          0
  unexpectedCfgMessage       0   unexpectedHashPayload      0

  authGood                   0   authFailed                 0
  xAuthNoSA                  0   xAuthNotEncrypted          0
  proccessXAuthCallbackNoSa  0   unsupportedAttribute       0

  ackRecv                    0   ackSent                    0
  repRecv                    0   repSent                    0
  reqRecv                    0   reqSent                    0
  setRecv                    0   setSent                    0
```

Table 45-31: Parameters in the output of the **show isakmp counter=transaction** command

Parameter	Meaning
msgLengthIncorrect	The number of messages received with an invalid message length.
tooManyAttributes	The number of unsupported number of attributes received.
hashNotFirst	The number of hash payloads that were not the first payload in a message.
hashInvalid	The number of invalid hash payloads received.
unexpectedCfgMessage	The number of configuration messages received when not expected.
authGood	The number of successful XAUTH authentications.
xAuthNoSA	The number of XAUTH messages received with no SA.
proccessXAuthCallbackNoSa	The number of Authentication callbacks with no SA.
ackRecv	The number of Acknowledge message received.
repRecv	The number of Reply messages received.
reqRecv	The number of Request messages received.
setRecv	The number of Set messages received.
reservedFieldNotZero	The number of messages received with a non-zero reserved field.
unexpectedAttributes	The number of messages received with unexpected attributes.
hashTooBig	The number of messages received with an unsupported hash length.
unexpectedMessage	The number of messages received when not expected.
unexpectedHashPayload	The number of hash payloads received when not expected.
authFailed	The number of failed attempts at XAUTH authentication.
xAuthNotEncrypted	The number of XAUTH messages received that were not encrypted as they should have been.
unsupportedAttribute	The number of messages received with an unsupported attribute.
ackSent	The number of Acknowledge messages sent.
repSent	The number of Reply messages sent.
reqSent	The number of Request messages sent.
setSent	The number of Set messages sent.

Figure 45-36: Example output from the **show isakmp counter=xde** command

```
ISAKMP Xdb Counters
                                    Good        Failed
  createXchg                           2             0
  deleteXchgById                       1             0
  getXchgById                         16             0
  getXchgByiCookie                     0             0
  getXchgByrCookie                     0             0
  getXchgByCookies                     1             0
  getXchgByMsgId                       0             0
  getXchgByMsgIdAndCookies             2             1
  getXchgByPolicy                      0             0
  getXchgByPeer                        0             1
  getXchgBySaIdSelectPol               0             0
```

Table 45-32: Parameters in the output of the **show isakmp counter=xde** command

Parameter	Meaning
createXchg	The number of successful and unsuccessful attempts to create an exchange in the ISAKMP exchange database.
deleteXchgById	The number of successful and unsuccessful attempts to remove an exchange from the ISAKMP exchange database.
getXchgById	The number of successful and unsuccessful attempts to get an exchange by ID from the ISAKMP exchange database.
getXchgByiCookie	The number of successful and unsuccessful attempts to retrieve an exchange by initiator cookie from the ISAKMP exchange database.
getXchgByrCookie	The number of successful and unsuccessful attempts to get an exchange by responder cookie from the ISAKMP exchange database.
getXchgByCookies	The number of successful and unsuccessful attempts to get an exchange by initiator and responder cookies from the ISAKMP exchange database.
getXchgByMsgId	The number of successful and unsuccessful attempts to get an exchange by message ID from the ISAKMP exchange database.
getXchgByMsgIdAndCookies	The number of successful and unsuccessful attempts to get an exchange by message ID and cookies from the ISAKMP exchange database.
getXchgByPolicy	The number of successful and unsuccessful attempts to get exchange by ISAKMP policy name from the ISAKMP exchange database.
getXchgByPeer	The number of successful and unsuccessful attempts to get exchange by peer IP address from the ISAKMP exchange database.
getXchgBySaIdSelectPol	The number of successful and unsuccessful attempts to get exchange by said, selector fields and policy name from the ISAKMP exchange database.

Examples To display the SPD and XPD counters, use the command:

```
sh isa cou=spd,xpd
```

To display all ISAKMP counters, use the command:

```
sh isa cou
```

Related Commands **show isakmp exchange**
show isakmp sa

show isakmp exchange

Syntax `SHow ISAkmp EXChange[=`*`exchange-id`*`]`

where *exchange-id* is a number from 0 to 4294967295

Description This command displays information about the specified or all current ISAKMP exchanges.

The **exchange** parameter specifies the identification number of the exchange to displayed. If a value is not specified, summary information about all current ISAKMP exchanges is displayed (Figure 45-37 on page 45-151, Table 45-33 on page 45-152). If a value is specified, detailed configuration and state information about the specified exchange is displayed (Figure 45-38 on page 45-154, Table 45-34 on page 45-155).

Figure 45-37: Example output from the **show isakmp exchange** command

```
ISAKMP Exchanges

   Id  Phase  State    PeerAddress               Type
   ------------------------------------------------------------
   12      1  KESENT   192.168.1.45              IDPROTECT
   13      2  WAIT_HSN 3ffe:2dde:3ff2:7777::3    QUICK
```

Table 45-33: Parameters in the output of the **show isakmp exchange** command

Parameter	Meaning
Id	The identification number used to identify the Exchange.
Phase	The current phase of the exchange; either 1, 1.5, or 2.
State	The current state of the exchange. For Main mode exchanges: IDLE SASENT SARECV KESENT KERECV AUTHSENT AUTHRECV UP For Quick mode exchanges: STARTING WAIT_HASH_SA_NONCE WAIT_HASH RECEIVING_MESSAGE SENDING_HASH_SA_NONCE SENDING_HASH DONE For Aggressive mode exchanges: IDLE SAKESENT SAKERECV SAKEAUTHSENT SAKEAUTHRECV AUTHSENT AUTHRECV UP For Transaction exchanges: IDLE REQSENT REQRECV REPSENT REPRECV SETSENT SETRECV ACKSENT ACKRECV UP For Informational exchanges, the State is IDLE.
Peer Address/Peer IP Address	The IP address of the peer for this exchange.

Table 45-33: Parameters in the output of the **show isakmp exchange** command (continued)

Parameter	Meaning
Type	The type of exchange: MAIN AGGRESSIVE TRANSACTION INFO QUICK

Figure 45-38: Example output from the **show isakmp exchange** command for a specific exchange in Main mode.

```
ISAKMP Exchange

  Id ..................................... 4
  Type ................................... MAIN
  State .................................. SASENT
  Phase .................................. 1
  Initiator .............................. TRUE
  DOI .................................... IPSEC
  Policy name ............................ main
  SA ..................................... 1
  Peer IP Address ........................ 202.36.163.201
  Local IP Address ....................... 202.36.163.161
  Encrypted .............................. FALSE
  Expecting message ...................... TRUE
  Has SA ................................. TRUE
  Initiator Cookie ....................... d464cc30b348efa7
  Responder Cookie ....................... 0000000000000000
  Message Id ............................. 00000000
  Set Commit bit ......................... FALSE
  Commit bit received .................... FALSE
  Send notifies .......................... TRUE
  Send deletes ........................... FALSE
  Message Retry Limit .................... 5
  Packet Retry Counter ................... 5
  Delete Delay Time (s) .................. 30
  Delete Delay Timer (time left (s)) .... 8
  Initial Message Retry Timeout (s) ..... 20
  Packet Retry Timer (time left(s)) ..... 10

Main Mode Status:

  ENCO Pass .............................. 0
  Shared Key Id .......................... 30
  Peer RSA Key Id ........................ 30
  Local RSA Key Id ....................... -
  Peer Certificate Id .................... -
  Local Certificate Id ................... -
  Local Policy
      Transform 0:
       Transform Number ............... 1
       Transform Id ................... 1
       Encryption Algorithm ........... DES - 56 bit
       Authentication Algorithm ...... SHA
       Authentication Method ......... PRESHARED
       Group Description .............. 768
       Group Type ..................... MODP
       Expiry Seconds ................. 86400
       Expiry KBytes .................. 1000
  Remote Policy

Sa Definition Information:
  Authentication Type ................. INVALID
  Encryption Algorithm ................ INVALID
  Hash Algorithm ...................... INVALID
  group Type .......................... INVALID
  group Description ................... INVALID
  DH Private Exponent Bits ............ 767
  expiry seconds ...................... 0
  expiry kilobytes .................... 0
```

Table 45-34: Parameters in the output of the **show isakmp exchange** command for a specific exchange

Parameter	Meaning
ISAKMP Exchange	**General information about the exchange**
Id	The identification number used to identify the Exchange.
Phase	The current phase of the exchange; either 1, 1.5, or 2.
State	The current state of the exchange. For Main mode exchanges: IDLE SASENT SARECV KESENT KERECV AUTHSENT AUTHRECV UP For Quick mode exchanges: STARTING WAIT_HASH_SA_NONCE WAIT_HASH RECEIVING_MESSAGE SENDING_HASH_SA_NONCE SENDING_HASH DONE For Aggressive mode exchanges: IDLE SAKESENT SAKERECV SAKEAUTHSENT SAKEAUTHRECV AUTHSENT AUTHRECV UP For Transaction exchanges: IDLE REQSENT REQRECV REPSENT REPRECV SETSENT SETRECV ACKSENT ACKRECV UP For informational exchanges, the State is IDLE. For completed exchanges awaiting deletion, the State is DELETEDELAY.

Table 45-34: Parameters in the output of the **show isakmp exchange** command for a specific exchange (continued)

Parameter	Meaning
Type	The type of exchange: MAIN AGGRESSIVE TRANSACTION INFO QUICK.
Initiator	Whether this router was the initiator of the exchange.
DOI	The domain of interpretation to be used by the exchange. Currently only IPsec is allowed.
Policy name	The name of the ISAKMP policy.
SA	The ID of the ISAKMP SA associated with this exchange.
Peer Address/Peer IP Address	The IP address of the peer for this exchange.
Local IP Address	The IP address of the local device for this exchange.
Encrypted	Whether the exchange is under the protection of an ISAKMP SA and is encrypting messages.
Expecting message	Whether the exchange is waiting for a new message.
Has SA	Whether the exchange has an ISAKMP SA associated with it.
Initiator Cookie	The 8-byte random number the initiator generates to identify the exchange.
Responder Cookie	The 8-byte random number the responder generates to identify the exchange.
Message Id	The 4-byte random number that identifies the exchange.
Set Commit bit	Whether the messages transmitted by this exchange set the commit bit.
Commit bit received	Whether messages received by this exchange set the commit bit.
Send notifies	Whether Notify payloads are sent with this exchange.
Send deletes	Whether Delete payloads are sent with this exchange.
Message Retry Limit	The number of times each message is retransmitted before the exchange expires.
Packet Retry Counter	The number retries left with the current sent message.
Delete Delay Time (s)	The number of seconds between the completion of the exchange and its deletion.
Delete Delay Timer (time left (s))	The number of seconds left before a completed exchange is deleted.
Initial Message Retry Timeout (s)	The number of seconds between the first message and the first retransmission.
Packet Retry Timer (time left(s))	The number of seconds left before next retransmission.
Main and Aggressive Mode	**Information about a Main mode or Aggressive mode exchange**
ENCO Pass	The number of passes to the ENCO processing unit.
Shared Key Id	The ENCO key identification number of the shared key used for PRESHARED authentication.

Table 45-34: Parameters in the output of the **show isakmp exchange** command for a specific exchange (continued)

Parameter	Meaning
Peer RSA Key Id	The ENCO key identification number of the RSA key used for RSAENCR authentication.
Local RSA Key Id	The ENCO ID of the key to be used as the local RSA key.
Peer Certificate Id	The ENCO ID of the certificate to be used as the peer's certificate.
Local Certificate Id	The ENCO ID of the certificate to be used as the local certificate.
Local Policy	A description of the local policy used in this exchange.
Remote Policy	A description of the local policy used in this exchange.
Transform	A description of the transform payload.
Transform Number	The number of the transform payload as it appears in an SA payload.
Transform Id	The ID for a transform payload.
Encryption Algorithm	The encryption algorithm to be used protect messages generated or received by this exchange.
Authentication Algorithm	The hash algorithm to be used to authenticate messages generated or received by this exchange.
Authentication Method	The method of authentication used in phase 1; either PRESHARED, RSAENCR, or RSASIG.
Group Description	The Diffie-Hellman group identification: 512 768 1024 INVALID
Group Type	The group type to perform Diffie-Hellman key exchange.
DH Private Exponent Bits	The length in bits of the DH private exponent.
Expiry Seconds	The expiry seconds as specified by local policy.
Expiry KBytes	The expiry kilobytes as specified by local policy.
Transaction Mode	**Information about a Quick mode exchange**
Mode	The mode of the transaction exchange; either XAUTH, SET, or REQ.
Id	An identifier to link separate exchanges together.
Quick Mode Status	**Information about a Quick mode exchange**
SA id	The ID of the ISAKMP Security Association this exchange is using.
local address	The local IP address of the exchange.
policy name	The name of the ISKAMP policy from which the SA this exchange is using was created
use PFS-KEY	Whether the exchange is using Perfect Forward Security for keys
use PFS-ID	Whether the exchange is using Perfect Forward Security for IDs
DH group ID	The ID of the Diffie-Hellman group being used by this exchange

Table 45-34: Parameters in the output of the **show isakmp exchange** command for a specific exchange (continued)

Parameter	Meaning
msg retry limit	The number of times the exchange tries to retransmit messages
msg retry timeout	The initial time period after which the exchange tries to retransmit a message
IPSEC exchange info	**IPsec information about the exchange**
IDi	Information about the initiator ID
type	The ID type
protocol Id	The protocol identifier of the ID
port	The port number of the ID
data	The ID data
IDr	Information about the responder ID
group Id	The IPsec group ID
Sa Definition Information:	**Information about the Security Association definition for the exchange**
Authentication Type	The method of authentication used for this SA.
Encryption Algorithm	The encryption algorithm used by this SA; either DES, 3DES2KEY, 3DESOUTER, or 3DESINNER.
Hash Algorithm	Whether to use the SHA or MD5 hash algorithm to authenticate data for this SA.
group Type	The Diffie-Hellman type used for key exchange; only MODP groups are supported.
group Description	The Diffie-Hellman group identification; either MODP512, MODP768, MODP1024, or INVALID.
expiry seconds	The negotiated expiry seconds for this SA.
expiry kilobytes	The negotiated expiry kilobytes for this SA.

Examples To display a list of all ISAKMP exchanges, use the command:

```
sh isa exc
```

Related Commands **show isakmp counters**
show isakmp sa

show isakmp policy

Syntax `SHow ISAkmp POLicy[=`*`name`*`]`

where *name* is a string up to 24 characters long. If the string contains spaces, it must be in double quotes.

Description This command displays information about the specified or all ISAKMP policies.

The **policy** parameter specifies the name of the IAKMP policy to be displayed. If a value is not specified, summary information for all ISAKMP policies is displayed (Figure 45-39 on page 45-159, Table 45-35 on page 45-159). If a value is specified, detailed information about the specified ISAKMP policy is displayed (Figure 45-40 on page 45-160, Table 45-36 on page 45-160). IPv6 addresses may be too long to display in the summary, in which case the full address can be seen by viewing the output for the particular policy.

Figure 45-39: Example output from the **show isakmp policy** command

```
Name                 PeerAddress
                     Mode    AuthType   EncAlg  HashAlg
-------------------------------------------------------------
my_isakmp_policy     192.168.2.1
                     ID_PROT PRESHARED  DES     SHA
```

Table 45-35: Parameters in the output of the **show isakmp policy** command

Parameter	Meaning
Name	The manager-assigned name for the ISAKMP policy.
PeerAdresss	The IP address of the peer for this policy, or ANY when the policy is used for remote peers whose IP address is not known at startup.
Mode	Whether the phase 1mode for the policy is IDPROT or AGGR.
AuthType	Whether the authentication type for the policy is PRESHARED, RSAENCR, or RSASIG.
EncAlg	The encryption algorithm to be used by the ISAKMP SAs created by this policy; either DES, 3DES2KEY, 3DESOUTER, or 3DESINNER.
HashAlg	Whether to use SHA or MD5 as the hash algorithm by the ISAKMP SAs created by this policy.

Figure 45-40: Example output from the **show isakmp policy** command for a specific policy.

```
ISAKMP Policy
  Name ................................ my_isakmp_policy
  Peer Address ........................ 202.36.163.201
  Phase1 Mode ......................... IDPROT
  Authentication Type ................. PRESHARED
  Extended Authentication ............. NONE
  Extended Authentication Type ........ -
  Extended Authentication User Name ..... -
  Extended Authentication Password ..... -
  Key Id .............................. 30
  Local RSA key ....................... -
  Peer Certificate Id ................. -
  Phase 2 Exchanges Limit ............. NONE
  PreNegotiate ........................ TRUE
  DOI ................................. IPSEC
  Send Notify Messages ................ TRUE
  Send Delete Messages ................ FALSE
  Always Send ID Messages ............. FALSE
  Commit Bit .......................... FALSE
  Message Retry Limit ................. 5
  Message Time Out .................... 20
  Exchange Delete Delay ............... 30
  Source Interface .................... -
  VPN Client Policy File Name ......... -
  Local ID ............................ -
  Remote ID ........................... IPv4:192.68.1.2
  DebugFlag ........................... 00000000

SA Specification
  Encryption Algorithm ................ DES - 56 bit
  Hash Algorithm ...................... SHA
  Group Description ................... 1
  DH Private Exponent Bits ............ 767
  Heartbeat Mode ...................... NONE
  Group Type .......................... MODP
  Expiry Seconds ...................... 86400
  Expiry Kilobytes .................... 1000
  NAT Traversal ....................... TRUE
```

Table 45-36: Parameters in the output of the **show isakmp policy** command for specific policy

Parameter	Meaning
Name	The manager-assigned name for the ISAKMP policy.
Peer Adresss	The IP address of the peer for this policy, or ANY when the policy is used for remote peers whose IP address is not known at startup.
Phase1 Mode	Whether the phase 1 mode for the policy is IDPROT or AGGR.
Authentication Type	Whether the authentication type for the policy is PRESHARED, RSAENCR, or RSASIG.
Extended Authentication	Whether the extended authentication (XAUTH) role for the router is CLIENT, SERVER, or NONE.
Extended Authentication Type	Whether the type of XAUTH is RADIUS_CHAP or GENERIC.

Table 45-36: Parameters in the output of the **show isakmp policy** command for specific policy (continued)

Parameter	Meaning
Extended Authentication User Name	The user name for XAUTH. Used when the router acts as an XAUTH client.
Extended Authentication Password	The password for XAUTH. Used when the router acts as an XAUTH client.
Key Id	The key identification number for the authentication key.
Local RSA key	The ID of the local RSA key used for authentication.
Peer Certificate Id	The ID of the certificate used to authenticate the peer.
Phase 2 Exchanges Limit	The number of phase 2 exchanges allowed before renegotiating a new ISAKMP SA.
PreNegotiate	Whether an ISAKMP SA is to be prenegotiated.
DOI	The Domain of Interpretation for the ISAKMP policy; only IPSEC is supported.
Send Notify Messages	Whether Notify messages are allowed.
Send Delete Messages	Whether Delete messages are allowed.
Always Send ID Messages	Whether ID messages are always sent when an ISAKMP SA is being negotiated.
Commit Bit	Whether the commit bit is set.
Message Retry Limit	The number of retries for each message before the exchange expires.
Message Time Out	The number of seconds before retranmission of the message.
Exchange Delete Delay (s)	The delay period, in seconds, between the completion and the deletion of ISAKMP exchanges notified for this policy.
Source Interface	The source interface the messages are sent to and received from.
VPN Client Policy File Name	Specifies the policy to be sent to remote ISAKMP peers when requested. For this feature to work, the **policyserverenabled** parameter on the **enable isakmp** command must be **true**. This parameter is for the AT-VPN Client Windows software package. For more information on this, see the AT-VPN Client documentation.
Local ID	The ID to be used when identifying the local device: IPv4: followed by an IP address IPv6: followed by an IPv6 address FQDN: followed by a fully-qualified domain name USER_FQDN: followed by a user fully-qualified domain name DN: followed by an X.500 distinguished name
Remote ID	The ID to be used when identifying the remote device: IPv4: followed by an IP address IPv6: followed by an IPv6 address FQDN: followed by a fully-qualified domain name USER_FQDN: followed by a user fully-qualified domain name DN: followed by an X.500 distinguished name

Table 45-36: Parameters in the output of the **show isakmp policy** command for specific policy (continued)

Parameter	Meaning
DebugFlag	A flag indicating the ISAKMP debugging options enabled.
SA Specification	Information about the Security Association specification for the policy.
Encryption Algorithm	The encryption algorithm to be used by the ISAKMP SAs created by this policy: AES - 128 bit AES - 192 bit AES - 256 bit DES - 56 bit 3DES - 112 bit -outer CBC 3DES - 168 bit -inner CBC
Hash Algorithm	Whether to use SHA or MD5 as the hash algorithm by the ISAKMP SAs created by this policy.
Group Description	Whether the Diffie-Hellman group number is 0, 1, or 2.
DH Private Exponent Bits	The length in bits of the DH private exponent.
Heartbeat Mode	Specifies whether the policy is configured to send or receive ISAKMP heartbeats; either NONE, SEND, RECEIVE, or BOTH.
Group Type	The Diffie-Hellman group type. Only MODP is supported.
Expiry Seconds	The lifetime in seconds for the ISAKMP SAs created by this policy.
Expiry Kilobytes	The maximum number of kilobytes of data that can be processed by the ISAKMP SAs created by this policy.
NAT Traversal	Whether NAT-T is enabled or disabled.

Examples To display a list of all ISAKMP policies, use the command:

```
sh isa pol
```

Related Commands **show isakmp**
show isakmp counters

show isakmp sa

Syntax `SHow ISAkmp SA[=sa-id]`

where *sa-id* is a number from 0 to 4294967295

Description This command displays information about all ISAKMP SAs or a specific one.

The **sa** parameter specifies the identification number of the SA for which information is to be displayed. If a value is not specified, summary information is displayed for all (Figure 45-41 on page 45-163, Table 45-37 on page 45-163). If a value is specified, detailed configuration and state information is displayed about the specific SA (Figure 45-42 on page 45-164, Table 45-38 on page 45-165).

Figure 45-41: Example output from the **show isakmp sa** command

```
SA Id PeerAddress     EncA.  HashA. Bytes                Seconds
-------------------------------------------------------------------------
    1 192.168.1.45    DES    SHA    100000/100000/88     86400/86400/21353
    2 202.124.2.68    DES    MD5    100000/100000/96     86400/86400/17353
```

Table 45-37: Parameters in the output of the **show isakmp sa** command

Parameter	Meaning
SA Id	The identification number for the SA.
PeerAddress	The IP address of the peer for this SA.
EncA	The encryption algorithm used by this SA: AES128 AES192 AES256 DES 3DES2KEY 3DESOUTER 3DESINNER
HashA	The hash algorithm used by this SA to authenticate data; either SHA or MD5.
Bytes	The expiry information in bytes. The first value is the hard expiry limit and displays the number of bytes of data that can be processed before the SA is destroyed. The second value is the soft expiry limit and displays the number of bytes of data that can be processed before an attempt is made to negotiate a new SA. The third value is the number of bytes of data already processed.
Seconds	The expiry information in seconds. The first value is the hard expiry limit after which the SA is destroyed. The second value is the soft expiry limit after which an attempt to negotiate a new SA is made. The third value is the number of seconds already used.

Figure 45-42: Example output from the **show isakmp sa** command for a specific SA.

```
 SA Id ................................ 1
  Initiator Cookie ..................... e418dba372510e53
  Responder Cookie ..................... 80c30ff4f2cb3f29
  DOI .................................. IPSEC
  Policy name .......................... main
  State ................................ ACTIVE
  Local address ........................ 202.36.163.161
  Remote Address ....................... 202.36.163.201
  Time of establishment ..............
  Commit bit set ....................... FALSE
  Send notifies ........................ TRUE
  Send deletes ......................... FALSE
  Message Retry Limit .................. 5
  Initial Message Retry Timeout (s) ... 20
  Exchange Delete Delay (s) ........... 30
  Do Xauth ............................. FALSE
    Xauth Finished ..................... TRUE
  Expiry Limit (bytes) ................. 1024000
  Soft Expiry Limit (bytes) ........... 896000
  Bytes seen ........................... 304
  Expiry Limit (seconds) ............... 86400
  Soft Expiry Limit (seconds) ......... 75600
  Seconds since creation ............... 2117
  Number of Phase 2 exchanges allowed . 4294967295
  Number of acquires queued ........... 0

Sa Definition Information:
  Authentication Type .................. PRESHARED
  Encryption Algorithm ................. DES - 56 bit
  Hash Algorithm ....................... SHA
  group Type ........................... MODP
  group Description .................... MODP768
  DH Private Exponent Bits ............. 767
  expiry seconds ....................... 86400
  expiry kilobytes ..................... 1000
XAuth Information:
  Id ................................... 0
  Next Message ......................... UNKNOWN
  Status ............................... FAIL
  Type ................................. Generic
  Max Failed Attempts .................. 0
  Failed Attempts ...................... 0
NAT-Traversal Information:
  NAT-T enabled ........................ YES
  Peer NAT-T capable ................... YES
  NAT discovered ....................... REMOTE
Heartbeat information:
  Send Heartbeats ...................... NO
  Next sequence number tx .............. 1
  Receive Heartbeats ................... NO
  Last sequence number rx .............. 0
```

Table 45-38: Parameters in the output of the **show isakmp sa** command for a specific SA.

Parameter	Meaning
SA Id	The identification number for the SA.
Initiator Cookie	An 8-byte random number generated by the initiator.
Responder Cookie	An 8-byte random number generated by the responder.
DOI	The domain of interpretation. Currently only IPSEC.
Policy name	The ISAKMP policy name.
State	State of the ISAKMP SA: ACTIVE EXPIRED DOING_NEW_GROUP DOING_XAUTH
Local Address	The local IP address for this SA.
Remote Address	The IP address of the peer for this SA.
Time of establishment	The time when the SA was created.
Commit bit set	Whether exchanges using this SA are committed.
Send notifies	Whether Notify messages are allowed using this SA.
Send deletes	Whether Delete messages are allowed using this SA.
Message Retry Limit	The number of times a message is resent when no reply is received.
Initial Message Retry Timeout(s)	The initial timeout, in seconds, before a message is resent.
Exchange Delete Delay (s)	The delay period, in seconds, between the completion and the deletion of ISAKMP exchanges over this SA.
Do Xauth	Whether this SA is authenticated using XAUTH.
Xauth Finished	Whether XAUTH authentication has been completed.
Expiry Limit (bytes)	The number of bytes processed by this SA before it is deleted.
Soft Expiry Limit (bytes)	The number of bytes processed by this SA before it is renegotiated.
Bytes seen	The number of bytes processed by this SA to date.
Expiry Limit (seconds)	The time before this SA is deleted.
Soft Expiry Limit (seconds)	The time before this SA is renegotiated.
Seconds since creation	The number of seconds since this SA was created.
Number of Phase 2 exchanges allowed	The maximum number of phase 2 exchanges allowed over this SA.
Number of acquires queued	The number of IPSEC acquire requests waiting to be processed.
Sa Definition Information	Information about the Security Associations definition
Authentication Type	The method of authentication used for this SA.

Table 45-38: Parameters in the output of the **show isakmp sa** command for a specific SA. (continued)

Parameter	Meaning
Encryption Algorithm	The encryption algorithm used by this SA: AES128 AES192 AES256 DES 3DES2KEY 3DESOUTER 3DESINNER
Hash Algorithm	The hash algorithm used by this SA to authenticate data; either SHA or MD5.
group Type	The Diffie-Hellman type used for key exchange; only MODP groups are supported.
group Description	The Diffie-Hellman group identification: MODP512 MODP768 MODP1024 INVALID
DH Private Exponent Bits	The length in bits of the DH private exponent.
expiry seconds	The negotiated expiry seconds for this SA.
expiry kilobytes	The negotiated expiry kilobytes for this SA.
XAuth Information	Information about Extended Authentication
Id	The ID used to identify messages from an XAUTH authentication.
Next Message	Whether the next XAUTH message to be sent is REQ or SET.
Status	Whether XAUTH authentication is okay or has failed.
Type	Whether the type of XAUTH authentication is generic or RADIUS-CHAP.
Max Failed Attempts	The maximum number of failed XAUTH attempts before the SA is deleted.
Failed Attempts	The current number of failed XAUTH attempts.
NAT-Traversal Information	Information about NAT-T capability.
NAT-T enabled	Whether NAT-T is enabled on the router.
Peer NAT-T capable	Whether the remote peer sent a valid NAT-T vendor ID.
NAT discovered	Whether a NAT device has been detected between peers: No - Not detected Remote - Detected at the remote site Local - Detected at this site Both - Local and remote Unknown -The peer is not NAT-T capable or the NAT discovery process is incomplete
Heartbeat Information	Information about ISAKMP heartbeats
Send Heartbeats	Whether the ISAKMP SA is configured to send ISAKMP heartbeats.

Table 45-38: Parameters in the output of the **show isakmp sa** command for a specific SA. (continued)

Parameter	Meaning
Next sequence number tx	The sequence number to be used in the next heartbeats sent.
Receive Heartbeats	Whether the ISAKMP SA is configured to receive ISAKMP heartbeats.
Last sequence number rx	The sequence number received in the last heartbeats.

Examples To display a list of all ISAKMP SAs, use the command:

```
sh isa sa
```

Related Commands **show isakmp counters**
show isakmp exchange

show sa

Syntax `SHow SA[={`*`sa-id`*`|ALL}]`

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

Description This command displays information about existing security associations. This command configures pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration.

If a security association is not specified, a summary of all security associations is displayed (Figure 45-43 on page 45-167, Table 45-39 on page 45-168).

If a security association is specified, detailed information about the specified security association is displayed (Figure 45-44 on page 45-168, Table 45-40 on page 45-169).

If **all** is specified, detailed information about all security association is displayed.

Figure 45-43: Example output from the **show sa** command

```
Summary SA output

ID   Dir   Expires         SPI  Process
-------------------------------------------------------------
  1  Both  None           1000
-------------------------------------------------------------
```

Table 45-39: Parameters in the output of the **show sa** command

Parameter	Meaning
ID	The identification number of the security association.
Dir	Whether the direction of the security association is in, out, or both.
Expires	The expiry date of the security association, if any.
SPI	The Security Parameters Index of the security association.
Comp	Whether compression is enabled on this SA.
Encryption/key	The encryption algorithm (either DES-CBC, 3DES-CBC, or NONE) and the identification number of the encryption key used by the security association.

Figure 45-44: Example output from the **show sa** command for a specific security association.

```
ID ..................... 1
SPI ................... 1000
User .................. IP
Expiry date ........... None
Direction ............. Both
Compression ........... OFF
Encryption ............ DES-CBC
  Key ID 1 ............ 1
Local members:
  202.36.163.21    255.255.255.255
Remote members:
  192.168.70.2     255.255.255.255

State:
Encode Channel ........ 0
Decode Channel ........ 0
Encode State .......... Attached
Decode State .......... Attached
```

Table 45-40: Parameters in the output of the **show sa** command for a specific security association

Parameter	Meaning
ID	The identification number of the security association.
SPI	The Security Parameters Index of the security association.
User	The user module attached to this SA.
Expiry date	The expiry date of the security association, if any.
Direction	Whether the direction of the security association is in, out, or both.
Compression	Whether compression is enabled on this SA.
Encryption	The encryption algorithm used by the security association: DES-CBC 3DES-112bit-OuterCBC 3DES-168bit-InnerCBC NONE
Key ID 1	The identification number of the encryption key used by the security association.
Local members	A list of ranges of IP addresses that are local members of the security association.
Remote members	A list of ranges of IP addresses that are remote members of the security association.
State:	The state of the SA's ENCO channels.
Encode Channel	The number of the ENCO encoding channel to which the SA is attached.
Decode Channel	The number of the ENCO decoding channel to which the SA is attached.
Encode State	The state of the ENCO encoding channel: Initial Attaching Attach Fail Attached
Decode State	The state of the ENCO decoding channel: Initial Attaching Attach Fail Attached

Related Commands

create sa
set sa
show sa counter

show sa counter

Syntax `SHow SA[=sa-id] COUnter`

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

Description This command displays information counters for the Security Association module. This command configures pre-IPsec SA functionality only. It is not used for IPsec or ISAKMP configuration.

If a security association is not specified, summary counters for all security associations are displayed (Figure 45-45 on page 45-170, Table 45-41 on page 45-171).

If a security association is specified, counters for the specified security association are displayed (Figure 45-46 on page 45-173, Table 45-42 on page 45-173).

Figure 45-45: Example output from the **show sa counter** command

```
SA Counters
 addrSearchBadIndex          0   addrSearchCacheHit          0
 addrSearchNotExist          0   addUserNotExist             0
 attachBadBPMode             0   attachGood                  0
 attachUserCreateFail        0   callBackBadIndex            0
 callBackNotExist            0   confBadBPMode               0
 confUserNotExist            0   decodeUserNotExist          0
 deletUserNotExist           0   detachGood                  0
 detachUserNotExist          0   encodeUserNotExist          0
 hasAddrDUserNotExist        0   ipAddAlreadyExists          0
 ipAddGood                   2   ipAddUnknownClass           0
 ipDeleteGood                0   ipDeleteNotFound            0
 processUserDecodeCAFailed   0    processUserDecodeNoAddr    0
 processUserDecodeNoSpi      0    processUserEncodeCAFailed  0
 processUserEncodeGood       0   processUserEncodeNoAddr     0
 secAssAddBadIndex           0   secAssAddNotExist           0
 secAssActBadIndex           0   secAssActNotExist           0
 secAssEncoHandBadIndex      0   secAssEncoHandNotExist      0
 secAssCreateAlreadyExist    0    secAssDeleteBadIndex       0
 secAssDeleteNotExist        0   secAssFindSpiFail           0
 secAssDestroyBadIndex       0   secAssDestroyNotExist       0
 secAssDestroyGood           0
 secAssFindSpiGood           0   secAssGetConfigBadIndex     0
 secAssGetConfigNotExist     1   secAssModifyBadIndex        0
 secAssModifyNotExist        0   userAddSecAssBadIndex       0
 userAddSecAssNotExist       0  userCreateGood               0
 userCreateNoSlot            0   userCreateUserExists        0
 userDelSecAssBadIndex       0   userDelSecAssNotExist       0
 userDestroyBadIndex         0   userDestroyGood             0
 userDestroyNotExist         0   userModifyBadIndex          0
 userModifyNotExist          0   userModifyGood              0
```

Table 45-41: Parameters in the output of the **show sa counter** command

Parameter	Meaning
addrSearchBadIndex	The number of address searches attempted with an invalid user.
addrSearchCacheHit	The number of address searches resolved by a cache hit.
addrSearchNotExist	The number of address searches attempted with an unused user.
addUserNotExist	The number of attempts to add an SA to a user that did not exist.
attachBadBPMode	The number of attach requests received from user modules with a bad BP mode specified.
attachGood	The number of successful attaches from user modules.
attachUserCreateFail	The number of attach requests that failed because the user object could not be created.
callBackBadIndex	The number of user call backs attempted that failed due to an invalid user id.
callBackNotExist	The number of user call backs attempted that failed due to an unused user.
confBadBPMode	The number of configure requests received from user modules with a bad BP mode specified.
confUserNotExist	The number of configure requests received from user modules for an invalid user.
decodeUserNotExist	The number of decode requests received from user modules for an invalid user.
deletUserNotExist	The number of attempts to delete an SA from a user that did not exist.
detachGood	The number of successful detaches by user modules.
detachUserNotExist	The number of detach attempts by user modules that failed because the user did not exist.
encodeUserNotExist	The number of encode requests received from user modules for an invalid user.
hasAddrDUserNotExist	The number of has address searches that failed due to an invalid user.
ipAddAlreadyExists	The number of add IP member requests that failed because the member already existed.
ipAddGood	The number of successful IP member adds.
ipAddUnknownClass	The number of add IP member requests that failed due to an unknown class.
ipDeleteGood	The number of successful IP member deletes.
ipDeleteNotFound	The number of delete IP member requests that failed because the member did not exist.
processUserDecodeCAFailed	The number of user decode requests that failed because an ENCO action failed.
processUserDecodeNoAddr	The number of user decode requests that failed because the address search failed.
processUserDecodeNoSpi	The number of user decode requests that failed because no SPI existed.
processUserEncodeCAFailed	The number of user encode requests that failed because an ENCO action failed.

Table 45-41: Parameters in the output of the **show sa counter** command (continued)

Parameter	Meaning
processUserEncodeGood	The number of successful user encode requests.
processUserEncodeNoAddr	The number of user encode requests that failed because the address search failed.
secAssAddBadIndex	The number of attempts to add an SA to a user that failed because the SA was invalid.
secAssAddNotExist	The number of attempts to add an SA to a user that failed because the SA did not exist.
secAssActBadIndex	The number of ENCO actions that failed because the SA was invalid.
secAssActNotExist	The number of ENCO actions that failed because the SA did not exist.
secAssEncoHandBadIndex	The number of ENCO actions that were returned with an invalid SA.
secAssEncoHandNotExist	The number of ENCO actions that were returned with an unused SA.
secAssCreateAlreadyExist	The number of create SA operations that failed because the SA already existed.
secAssDeleteBadIndex	The number of delete SA operations that failed because the SA was invalid.
secAssDeleteNotExist	The number of delete SA operations that failed because the SA did not exist.
secAssFindSpiFail	The number of find SPI operations that failed.
secAssDestroyBadIndex	The number of destroy SA operations that failed because the SA was invalid.
secAssDestroyNotExist	The number of destroy SA operations that failed because the SA did not exist.
secAssDestroyGood	The number of successful destroy SA operations.
secAssFindSpiGood	The number of successful find SPI operations.
secAssGetConfigBadIndex	The number of get configuration operations that failed because the SA was invalid.
secAssGetConfigNotExist	The number of get configuration operations that failed because the SA did not exist.
secAssModifyBadIndex	The number of modify SA operations that failed because the SA was invalid.
secAssModifyNotExist	The number of modify SA operations that failed because the SA did not exist.
userAddSecAssBadIndex	The number of add SA operations that failed because the SA was invalid.
userAddSecAssNotExist	The number of add SA operations that failed because the SA did not exist.
userCreateGood	The number of successful create user operations.
userCreateNoSlot	The number of create user operations that failed because there was no spare user slot.
userCreateUserExists	The number of create user operations that failed because the user already existed.

Table 45-41: Parameters in the output of the **show sa counter** command (continued)

Parameter	Meaning
userDelSecAssBadIndex	The number of delete SA operations that failed because the SA was invalid.
userDelSecAssNotExist	The number of delete SA operations that failed because the SA did not exist.
userDestroyBadIndex	The number of destroy user operations that failed because the SA was invalid.
userDestroyGood	The number of successful destroy user operations.
userDestroyNotExist	The number of destroy user operations that failed because the SA did not exist.
userModifyBadIndex	The number of modify user operations that failed because the SA did not exist.
userModifyNotExist	The number of modify user operations that failed because the SA was invalid.
userModifyGood	The number of successful modify user operations.

Figure 45-46: Example output from the **show sa counter** command for a specific security association.

```
addAlreadyExist              0   addGood                    0
addUserNotExist              0   deleteAlreadyIndex         0
deleteGood                   0   getConfigGood              0
encoCreateDAttachDead        0   encoCreateDAttachFail      1
encoCreateDAttachGood        0   encoCreateEAttachDead      0
encoCreateEAttachFail        1   encoCreateEAttachGood      0
encoModifyDReconfDead        0   encoModifyDReconfFail      0
encoModifyDReconfGood        0   encoModifyEReconfDead      0
encoModifyEReconfFail        0   encoModifyEReconfGood      0
encoActDecode                0   encoActDecodeNotAttached   0
encoActEncode                0   encoActEncodeNotAttached   0
encoActUnknownType           0   encoHandUnknownEvent       0
encoHandEncodeConfigured     0   encoHandEncodeConfigFail   1
encoHandEncodeChanAttached   0   encoHandEncodeDetached      0
encoHandEncoded              0   encoHandEncodeFail         0
encoHandDecodeConfigured     0   encoHandDecodeConfigFail   1
encoHandEncodeDiscard        0   encoHandDecodeDiscard      0
encoHandDecodeChanAttached   0   encoHandDecodeDetached      0
encoHandDecoded              0   encoHandDecodeFail         0
```

Table 45-42: Parameters in the output of the **show sa counter command** for a specific security association

Parameter	Meaning
addAlreadyExist	The number of requests to add a member to an SA that failed because the member already existed.
addGood	The number of successful requests to add a member to an SA.
addUserNotExist	The number of requests to add a user to an SA that failed because the user did not exist.
deleteAlreadyIndex	The number of delete member requests that failed because the member did not exist.

Table 45-42: Parameters in the output of the **show sa counter command** for a specific security association (continued)

Parameter	Meaning
deleteGood	The number of successful delete member requests.
getConfigGood	The number of successful get configuration operations.
encoCreateDAttachDead	The number of dead events detected while attaching a decoding channel to the ENCO module.
encoCreateDAttachFail	The number of unsuccessful attempts to attach a decoding channel to the ENCO module.
encoCreateDAttachGood	The number of successful attempts to attach a decoding channel to the ENCO module.
encoCreateEAttachDead	The number of dead events detected while attaching a encoding channel to the ENCO module.
encoCreateEAttachFail	The number of unsuccessful attempts to attach a encoding channel to the ENCO module.
encoCreateEAttachGood	The number of successful attempts to attach a encoding channel to the ENCO module.
encoModifyDReconfDead	The number of dead events detected while modifying a decoding channel to the ENCO module.
encoModifyDReconfFail	The number of unsuccessful attempts to modify a decoding channel to the ENCO module.
encoModifyDReconfGood	The number of successful attempts to modify a decoding channel to the ENCO module.
encoModifyEReconfDead	The number of dead events detected while modifying a encoding channel to the ENCO module.
encoModifyEReconfFail	The number of unsuccessful attempts to modify a encoding channel to the ENCO module.
encoModifyEReconfGood	The number of successful attempts to modify a encoding channel to the ENCO module.
encoActDecode	The number of decode ENCO actions sent.
encoActDecodeNotAttached	The number of decode ENCO actions not sent because the SA was not attached to the ENCO module.
encoActEncode	The number of encode ENCO actions sent.
encoActEncodeNotAttached	The number of encode ENCO actions not sent because the SA was not attached to the ENCO module.
encoActUnknownType	The number of unknown ENCO actions sent.
encoHandUnknownEvent	The number of unknown actions returned from the ENCO module.
encoHandEncodeConfigured	The number of good encode channel configure ENCO actions.
encoHandEncodeConfigFail	The number of bad encode channel attach ENCO actions.
encoHandEncodeChanAttached	The number of good encode channel attach ENCO actions.
encoHandEncodeDetached	The number of good encode channel detach ENCO actions.
encoHandEncoded	The number of good encode ENCO actions.
encoHandEncodeFail	The number of bad encode ENCO actions.

Table 45-42: Parameters in the output of the **show sa counter command** for a specific security association (continued)

Parameter	Meaning
encoHandDecodeConfigured	The number of good decode channel configure ENCO actions.
encoHandDecodeConfigFail	The number of decode channel attach ENCO actions.
encoHandEncodeDiscard	The number of destroy encode ENCO actions.
encoHandDecodeDiscard	The number of destroy decode ENCO actions.
encoHandDecodeChanAttached	The number of good decode channel attach ENCO actions.
encoHandDecodeDetached	The number of good decode channel detach ENCO actions.
encoHandDecoded	The number of good decode ENCO actions.
encoHandDecodeFail	The number of bad decode ENCO actions.

Related Commands

create sa
set sa
show sa

show sa user

Syntax

```
SHow SA USER
```

Description

This command displays information counters about user objects of the Security Association (SA) module (Figure 45-47 on page 45-175, Table 45-43 on page 45-176).

This command configures pre-IPsec old functionality only. It is not used for IPsec or ISAKMP configuration.

Figure 45-47: Example output from the **show sa user** command

```
IP User:    1
 addGood                    1   addrSearchFail             2
 addrSearchGood             0   addSecAssNotExist          0
 callBackBlock              0   callBackDecode             0
 callBackDecoded            0   callBackDecodeFail         0
 callBackEncode             0   callBackEncoded            0
 callBackEncodeFail         0   callBackIndFailed          0
 callBackUnknownType        0   decodeNoAddress            0
 decodeNoBuffer             0
 decodeReceiveGood          0   deleteGood                 0
 deleteSecAssNotExist       0   encodeNoAddress            0
 encodeNoBuffer             0   hasAddrDNoAddress          0
 addSecAssAlreadyPresent    0   addSecAssGood              1
 addSecAssNotExistSecAss    0   delSecAssAlreadyPresent    0
 delSecAssGood              0   encodeReceiveGood          0
```

Table 45-43: Parameters in the output of the **show sa user** command

Parameter	Meaning
addGood	The number of good add user requests.
addrSearchFail	The number of unsuccessful address searches.
addrSearchGood	The number of successful address searches.
addSecAssNotExist	The number of requests to add a SA to the user that failed because the SA did not exist.
callBackBlock	The number of user packets that were blocked.
callBackDecode	The number of packets received to decode.
callBackDecoded	The number of packets successfully decoded.
callBackDecodeFail	The number of packets unsuccessfully decoded.
callBackEncode	The number of packets received to encode.
callBackEncoded	The number of packets successfully encoded.
callBackEncodeFail	The number of packets unsuccessfully encoded.
callBackIndFailed	The number of user operations that failed.
callBackUnknownType	The number of unknown user call back events.
decodeNoAddress	The number of decode packet requests received from the user with no address.
decodeNoBuffer	The number of decode packet requests received from the user with no buffer.
decodeReceiveGood	The number of good decode packet requests received from the user.
deleteGood	The number of good delete requests.
deleteSecAssNotExist	The number of requests to delete an SA from the user that failed because the SA did not exist.
encodeNoAddress	The number of encode packet requests received from the user with no address.
encodeNoBuffer	The number of encode packet requests received from the user with no buffer.
hasAddrDNoAddress	The number of has address operations that failed.
addSecAssAlreadyPresent	The number of add SA requests that failed because the SA was already present.
addSecAssGood	The number of good add SA requests received from the user.
addSecAssNotExistSecAss	The number of add SA requests that failed because the SA did not exist.
delSecAssAlreadyPresent	The number of delete SA requests that failed because the SA was not already present.
delSecAssGood	The number of good delete SA requests received from the user.
encodeReceiveGood	The number of good encode packet requests received from the user.

Related Commands

create sa
set sa
show sa
show sa counter