

Chapter 30

Security Services Configuration Guidelines

To configure security services, include statements at the [edit security] hierarchy level:

```
[edit security]
certificates {
  cache-size bytes;
  cache-timeout-negative seconds;
  certification-authority ca-profile-name {
    ca-name ca-identity;
    encoding (binary | pem);
    crl file-name;
    file certificate-file-name;
    enrollment-url url-name;
    ldap-url url-name;
  }
  enrollment-retry number;
  local certificate-name;
  maximum-certificates number;
  path-length bytes;
}
ike {
  proposal ike-proposal-name {
    authentication-algorithm (md5 | sha1);
    authentication-method (dsa-signatures | pre-shared-keys | rsa-signatures);
    dh-group (group1 | group2);
    encryption-algorithm (3des-cbc | des-cbc);
    lifetime-seconds seconds;
  }
  policy ike-peer-address {
    description policy-description;
    encoding (binary | pem);
    identity identity-name;
    local-certificate certificate-file-name;
    local-key-pair private-public-key-file;
    mode (aggressive | main);
    pre-shared-key (ascii-text key | hexadecimal key);
    proposal [ike-proposal-names];
  }
}
```

```

ipsec {
  proposal ipsec-proposal-name {
    authentication-algorithm (hmac-md5-96 | hmac-sha1-96);
    encryption-algorithm (3des-cbc | des-cbc);
    lifetime-seconds seconds;
    protocol (ah | esp | bundle);
  }
  policy ipsec-policy-name {
    perfect-forward-secrecy {
      keys (group1 | group2);
    }
    proposal [ ipsec-proposal-names ];
  }
  security-association name {
    mode (tunnel | transport);
    manual {
      direction (inbound | outbound | bi-directional) {
        auxiliary-spi auxiliary-spi;
        spi spi-value;
        protocol (ah | esp | bundle);
        authentication {
          algorithm (hmac-md5-96 | hmac-sha1-96);
          key (ascii-text key | hexadecimal key);
        }
        encryption {
          algorithm (des-cbc | 3des-cbc);
          key (ascii-text key | hexadecimal key);
        }
      }
    }
    dynamic {
      replay-window-size (32 | 64);
      ipsec-policy policy-name;
    }
  }
  traceoptions {
    file filename <files number> < size size>;
    flag all;
    flag database;
    flag general;
    flag ike;
    flag parse;
    flag policy-manager;
    flag routing-socket;
    flag timer;
  }
}

```



Note

Most of the configuration statements do not have default values. If you do not specify an identifier for a statement that does not have a default value, you cannot commit the configuration.

For information about IPSec monitoring and troubleshooting, see the *JUNOS Internet Software Configuration Guide: Operational Mode Command Reference*

This chapter describes the following tasks for configuring Internet Protocol Security (IPSec) and the Internet Key Exchange (IKE):

- Minimum Manual SA Configuration on page 413
- Minimum IKE Configuration on page 414
- Minimum Digital Certificates Configuration for IKE on page 415
- Configure Security Associations on page 416
- Configure an IKE Proposal (Dynamic SAs Only) on page 425
- Configure an IKE Policy for Preshared Keys on page 428
- Configure an IPSec Proposal on page 431
- Configure an IPSec Policy on page 433
- Digital Certificates Guidelines on page 436
- Configure Trace Options on page 446
- Configure the ES PIC on page 447
- Configure Traffic on page 448
- Configure an ES Tunnel Interface for a Layer 3 VPN on page 453
- JUNOScript XNM-SSL Service on page 453

Minimum Manual SA Configuration

To define a manual security associations (SA) configuration, you must include at least the following statements at the [edit security ipsec] hierarchy level.

```
[edit security ipsec]
security-association name {
  manual {
    direction (inbound | outbound | bi-directional) {
      spi spi-value;
      protocol (ah | esp | bundle);
      authentication {
        algorithm (hmac-md5-96 | hmac-sha1-96);
        key (ascii-text key | hexadecimal key);
      }
      encryption {
        algorithm (des-cbc | 3des-cbc);
        key (ascii-text key | hexadecimal key);
      }
    }
  }
}
```

Minimum IKE Configuration

To define a IKE configuration, include at least the following statements at the [edit security] hierarchy level.

```
[edit security]
ike {
  proposal ike-proposal-name {
    authentication-algorithm (md5 | sha1);
    authentication-method pre-shared-keys;
    dh-group (group1 | group2);
    encryption-algorithm (3des-cbc | des-cbc);
  }
  policy ike-peer-address {
    proposal [ike-proposal-names];
    pre-shared-key (ascii-text key | hexadecimal key);
  }
}
ipsec {
  policy ipsec-policy-name {
    proposal [ipsec-proposal-names];
  }
  proposal ipsec-proposal-name {
    authentication-algorithm (hmac-md5-96 | hmac-sha1-96);
    encryption-algorithm (3des-cbc | des-cbc);
    protocol (ah | esp | bundle);
  }
  security-association name {
    dynamic {
      ipsec-policy policy-name;
    }
  }
}
```

Minimum Digital Certificates Configuration for IKE

To define digital certificates configuration for IKE, include at least the following statements at the [edit security] hierarchy level.

```
[edit security certificates]
certification-authority ca-profile-name {
  ca-name ca-identity;
  crl file-name;
  enrollment-url url-name;
  file certificate-file-name;
  ldap-url url-name;
}
ike {
  proposal ike-proposal-name {
    authentication-algorithm (md5 | sha1);
    authentication-method pre-shared-keys;
    dh-group (group1 | group2);
    encryption-algorithm (3des-cbc | des-cbc);
  }
  policy ike-peer-address {
    identity identity-name;
    local-certificate certificate-file-name;
    local-key-pair private-public-key-file;
    proposal [ ike-proposal-names ];
  }
}
ipsec {
  policy ipsec-policy-name {
    proposal [ ipsec-proposal-names ];
  }
  proposal ipsec-proposal-name {
    authentication-algorithm (hmac-md5-96 | hmac-sha1-96);
    encryption-algorithm (3des-cbc | des-cbc);
    protocol (ah | esp | bundle);
  }
  security-association name {
    dynamic {
      ipsec-policy policy-name;
    }
  }
}
```

Configure Security Associations

To use IPsec security services, you create a security association (SA) between hosts. An SA is a simplex connection that allows two hosts to communicate with each other securely by means of IPsec. You can configure two types of SAs:

Manual—Requires no negotiation; all values, including the keys, are static and specified in the configuration. As a result, each peer must have the same configured options for communication to take place. For information about how to configure a manual SA, see “Configure Manual Security Associations” on page 419.

Dynamic—Specifies proposals to be negotiated with the tunnel peer. The keys are generated as part of the negotiation and therefore do not need to be specified in the configuration. The dynamic SA includes one or more proposal statements, which allow you to prioritize a list of protocols and algorithms to be negotiated with the peer. For information about how to configure a dynamic SA, see “Configure the ES PIC” on page 447.



Note

The JUNOS software does not perform a commit check when an SA name referenced in the BGP protocol section is not configured at the [edit security ipsec] hierarchy level.

We recommend that you configure no more than 512 dynamic security associations per ES PIC.

To configure an SA for IPsec, include the security-association statement and specify a security association name at the [edit security ipsec] hierarchy level:

```
[edit security ipsec]
security-association name;
```

This section describes the following topics related to configuring security associations:

Configure IPsec Mode on page 417

Configure Manual Security Associations on page 419

Configure Dynamic Security Associations on page 423

Configure IPsec Mode

The JUNOS software implementation of IPsec supports two modes of security: transport and tunnel mode. By default, tunnel mode is enabled.

This section discusses the following topics:

Transport Mode on page 417

Tunnel Mode on page 418

Transport Mode

In transport mode, the data portion of the IP packet is encrypted, but the IP header is not. Transport mode can be used only when the communication endpoint and cryptographic endpoint are the same. VPN gateways that provide encryption and decryption services for protected hosts cannot use transport mode for protected VPN communications. SAs are manually configured and static values must be configured on both ends of the SA.



Note

When you use transport mode, the JUNOS software supports only BGP for manual SAs.

To configure IPsec security for transport mode, include the mode statement and specify transport at the [edit security ipsec security-association *name*] hierarchy level:

```
[edit security ipsec security-association name]  
mode transport;
```

To apply tunnel mode, you configure manual SAs in transport mode and then reference the SA by name at the [edit protocols bgp] hierarchy level to protect a session with a given peer. For more information about how to reference the configured SA, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.



Note

You can configure BGP to peer over encrypted tunnels.

Tunnel Mode

You use tunnel mode when you use preshared keys with IKE to authenticate peers or digital certificates with IKE to authenticate peers. With you use preshared keys, you manually configure a preshared key, which must match that of its peer. With digital certificates, each router is dynamically or manually enrolled with a certification authority (CA). When a tunnel is established, the public keys used for IPsec are dynamically obtained through IKE and validated against the CA certificate. This avoids the manual configuration of keys on routers within the topology. Adding a new router to the topology does not require any security configuration changes to existing routers. Because of the high speeds of the transit interfaces, tunnel mode requires an ES PICs.

To configure the IPsec in tunnel mode, include the mode statement and specify tunnel at the [edit security ipsec security-association *name*] hierarchy level:

```
[edit security ipsec security-association name]  
mode tunnel;
```



Note

Tunnel mode requires the ES PIC.

In transport mode, the JUNOS software does not support authentication header (AH) or encapsulating security payload (ESP) header bundles.

The JUNOS software supports only BGP in transport mode.

To enable tunnel mode, follow steps in these sections:

1. Configure an IKE Proposal (Dynamic SAs Only) on page 425
2. Configure Security Associations on page 416
3. Configure the ES PIC on page 447
4. Configure Traffic on page 448

For more information about the ES PIC, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service* .

Configure Manual Security Associations

Manual SAs require no negotiation; all values, including the keys, are static and specified in the configuration. As a result, each peer must have the same configured options for communication to take place.

To configure the manual IPSec security association, include statements at the [edit security ipsec security-association *name* manual] hierarchy level:

```
[edit security ipsec security-association name manual]
direction (inbound | outbound | bi-directional) {
  auxiliary-spi auxiliary-spi-value;
  spi spi-value;
  protocol (ah | esp | bundle);
  authentication {
    algorithm (hmac-md5-96 | hmac-sha1-96);
    key (ascii-text key | hexadecimal key);
  }
  encryption {
    algorithm (des-cbc | 3des-cbc);
    key (ascii-text key | hexadecimal key);
  }
}
```

To configure manual SA statements, do the following:

Configure Direction on page 419

Configure the Protocol on page 420

Configure a Security Parameter Index (SPI) on page 421

Configure the Auxiliary Security Parameter Index on page 421

Configure Authentication on page 422

Configure Encryption on page 422

Configure Direction

The direction statement sets inbound and outbound IPSec processing. If you want to define different algorithms, keys, or security parameter index (SPI) values for each direction, you configure the inbound and outbound options. If you want the same attributes in both directions, use the bidirectional option.

To configure the direction of IPSec processing, include the direction statement and specify the direction at the [edit security ipsec security-association *name* manual] hierarchy level:

```
[edit security ipsec security-association name manual]
direction (inbound | outbound | bidirectional);
```

Example: Configure Inbound and Outbound Direction Statements

Define different algorithms, keys, and security parameter index values for each direction.

```
[edit security ipsec security-association name]
manual {
  direction inbound {
    protocol esp;
    spi 16384;
    encryption {
      algorithm 3des-cbc;
      key ascii-text 23456789012345678901234;
    }
  }
  direction outbound {
    protocol esp;
    spi 24576;
    encryption {
      algorithm 3des-cbc;
      key ascii-text 12345678901234567890abcd;
    }
  }
}
```

Example: Configure Bidirectional Statement

Define the same algorithms, keys, and security parameter index values for each direction.

```
[edit security ipsec security-association name manual]
direction bidirectional {
  protocol ah;
  spi 20001;
  authentication {
    algorithm hmac-md5-96;
    key ascii-text 123456789012abcd;
  }
}
```

Configure the Protocol

IPSec uses two protocols to protect IP traffic: ESP and AH. For transport mode SAs, both ESP and AH are supported. The AH protocol is used for strong authentication. The bundle option uses AH authentication and ESP encryption; it does not use ESP authentication because AH provides stronger authentication of IP packets.

To configure the IPSec protocol, include the protocol statement and specify ah, esp, or bundle at the [edit security ipsec security-association *name* manual direction (inbound | outbound | bidirectional) hierarchy level:

```
[edit security ipsec security-association name manual direction (inbound | outbound |
bi-directional)]
protocol (ah | bundle | esp);
```

Configure a Security Parameter Index (SPI)

An SPI is an arbitrary value that uniquely identifies which SA to use at the receiving host. The sending host uses the SPI to identify and select which SA to use to secure every packet. The receiving host uses the SPI to identify and select the encryption algorithm and key used to decrypt packets.



Note

Each manual SA must have a unique SPI and protocol combination.

Use the auxiliary SPI when you configure the protocol statement to use the bundle option.

To configure the SPI, include the `spi` statement and specify a value (256 through 16,639) at the [edit security ipsec security-association *name* manual direction (inbound | outbound | bi-directional)] hierarchy level:

```
[edit security ipsec security-association name manual direction (inbound | outbound |
bidirectional)]
spi spi-value;
```

Configure the Auxiliary Security Parameter Index

Use the auxiliary SPI when you configure the protocol statement to use the bundle option.



Note

Each manual SA must have a unique SPI and protocol combination.

To configure the auxiliary SPI, include the `auxiliary-spi` statement and specify a value (256 through 16,639) at the [edit security ipsec security-association *name* manual direction (inbound | outbound | bi-directional)] hierarchy level:

```
[edit security ipsec security-association name manual direction (inbound | outbound |
bidirectional)]
auxiliary-spi auxiliary-spi-value;
```

Configure Authentication

To configure an authentication algorithm, include the authentication statement and specify an authentication algorithm and a key at the [edit security ipsec security-association *name* manual direction (inbound | outbound | bi-directional)] hierarchy level:

```
[edit security ipsec security-association name manual direction (inbound | outbound |
bidirectional)]
authentication {
  algorithm (hmac-md5-96 | hmac-sha1-96);
  key (ascii-text key | hexadecimal key);
}
```

The algorithm can be one of the following:

hmac-md5-96—Hash algorithm that authenticates packet data. It produces a 128-bit authenticator value and 96-bit digest.

hmac-sha1-96—Hash algorithm that authenticates packet data. It produces a 160-bit authenticator value and a 96-bit digest.

The key can be one of the following:

ascii-text—ASCII text key. With the **hmac-md5-96** option, the key contains 16 ASCII characters. With the **hmac-sha1-96** option, the key contains 20 ASCII characters.

hexadecimal—Hexadecimal key. With the **hmac-md5-96** option, the key contains 32 hexadecimal characters. With the **hmac-sha1-96** option, the key contains 40 hexadecimal characters.

Configure Encryption

To configure IPSec encryption, include the encryption statement and specify an algorithm and key at the [edit security ipsec security-association *name* manual direction (inbound | outbound | bi-directional)] hierarchy level:

```
[edit security ipsec security-association name manual direction (inbound | outbound |
bi-directional)]
encryption {
  algorithm (des-cbc | 3des-cbc);
  key (ascii-text key | hexadecimal key);
}
```

The algorithm can be one of the following:

des-cbc—Encryption algorithm that has a block size of 8 bytes; its key size is 64 bits long.

3des-cbc—Encryption algorithm that has a block size of 24 bytes; its key size is 192 bits long.

**Note**

For a list of DES (Data Encryption Standard) encryption algorithm Weak and Semi-Weak keys, see RFC 2409.

For 3des-cbc, we recommend that the first 8 bytes are not the same as the second 8 bytes, and the second 8 bytes are the same as the third 8 bytes.

The key can be one of the following:

ascii-text—ASCII text key. With the des-cbc option, the key contains 8 ASCII characters. With the 3des-cbc option, the key contains 24 ASCII characters.

hexadecimal—Hexadecimal key. With the des-cbc option, the key contains 16 hexadecimal characters. With the 3des-cbc option, the key contains 48 hexadecimal characters.

**Note**

You cannot configure encryption when you use authentication header protocol (AH).

Configure Dynamic Security Associations

You configure dynamic SAs with a set of proposals that are negotiated by the security gateways. The keys are generated as part of the negotiation and therefore do not need to be specified in the configuration. The dynamic SA includes one or more proposals, which allow you to prioritize a list of protocols and algorithms to be negotiated with the peer.

To enable a dynamic SA, follow these steps:

1. Configure IKE proposals and IKE policies associated with these proposals.
2. Configure IPSec proposals and an IPSec policy associated with these proposals.
3. Associate an SA with an IPSec policy.

For more information about IKE policies and proposals, see “Configure an IKE Policy for Preshared Keys” on page 428 and “Configure an IKE Proposal (Dynamic SAs Only)” on page 425. For more information about IPSec policies and proposals, see “Configure an IPSec Policy” on page 433.

**Note**

Dynamic tunnel SAs require an ES PIC.

To configure a dynamic SA, include the dynamic statement and specify a 32- or 64-packet replay window size and an IPsec policy name at the [edit security ipsec security-association *name*] hierarchy level. The window-replay-size statement is optional.

```
[edit security ipsec security-association name]  
dynamic {  
  replay-window-size (32 | 64);  
  ipsec-policy policy-name;  
}
```



Note

If you want to establish a dynamic SA, the attributes in at least one configured IPsec and IKE proposal must match those of its peer.

The replay window is not used with manual SAs.

Configure an IKE Proposal (Dynamic SAs Only)

Dynamic SAs require Internet Key Exchange (IKE) configuration. With dynamic SAs, you configure IKE first, and then the SA. IKE creates dynamic SAs; it negotiates SAs for IPSec. The IKE configuration defines the algorithms and keys used to establish the secure IKE connection with the peer security gateway.

You can configure one or more IKE proposals. Each proposal is a list of IKE attributes to protect the IKE connection between the IKE host and its peer.

To configure an IKE proposal, include the proposal statement and specify a name at the [edit security ike] hierarchy level:

```
[edit security ike]
proposal ike-proposal-name;
authentication-algorithm (md5 | sha1);
authentication-method pre-shared-keys;
dh-group (group1 | group2);
encryption-algorithm (3des-cbc | des-cbc);
lifetime-seconds seconds;
```

To define proposal properties, include one or more of the following statements at the [edit security ike proposal *ike-proposal-name*] hierarchy level:

```
[edit security ike proposal ike-proposal-name]
authentication-algorithm (md5 | sha1);
authentication-method pre-shared-keys;
dh-group (group1 | group2);
encryption-algorithm (3des-cbc | des-cbc);
lifetime-seconds seconds;
```

This section discusses the following topics:

Configure an IKE Authentication Algorithm on page 426

Configure an IKE Authentication Method on page 426

Configure an IKE Diffie-Hellman Group on page 426

Configure an IKE Encryption Algorithm on page 427

Configure an IKE Lifetime on page 427

Example: Configure an IKE Proposal on page 427

Configure an IKE Authentication Algorithm

To configure an IKE authentication algorithm, include the authentication-algorithm statement at the [edit security ike proposal *ike-proposal-name*] hierarchy level:

```
[edit security ike proposal ike-proposal-name]  
authentication-algorithm (md5 | sha1);
```

The authentication algorithm can be one of the following:

md5—Produces a 128-bit digest.

sha1—Produces a 160-bit digest.

Configure an IKE Authentication Method

To configure an IKE authentication method, include the authentication-method statement at the [edit security ike proposal *ike-proposal-name*] hierarchy level:

```
[edit security ike proposal ike-proposal-name]  
authentication-method (dsa-signatures | pre-shared-keys | rsa-signatures);
```

The authentication method can be one of the following:

dsa-signatures—Digital Signature Algorithm

pre-shared-keys—Preshared keys.

rsa-signatures—RSA signatures.

Configure an IKE Diffie-Hellman Group

Diffie-Hellman is a public-key cryptography scheme that allows two parties to establish a shared secret over an insecure communications channel. It is also used within IKE to establish session keys.

To configure an IKE Diffie-Hellman group, include the dh-group statement at the [edit security ike proposal *ike-proposal-name*] hierarchy level:

```
[edit security ike proposal ike-proposal-name]  
dh-group (group1 | group2);
```

The group can be one of the following:

group1—Specifies that IKE use the 768-bit Diffie-Hellman prime modulus group when performing the new Diffie-Hellman exchange.

group2—Specifies that IKE use the 1,024-bit Diffie-Hellman prime modulus group when performing the new Diffie-Hellman exchange.

group2 provides more security but requires more processing time.

Configure an IKE Encryption Algorithm

To configure an IKE encryption algorithm, include the encryption-algorithm statement at the [edit security ike proposal *ike-proposal-name*] hierarchy level.

```
[edit security ike proposal ike-proposal-name]  
encryption-algorithm (3des-cbc | des-cbc);
```

The encryption algorithm can be one of the following:

3des-cbc—Encryption algorithm that has a key size of 24 bytes; its key size is 192 bits long.

des-cbc—Encryption algorithm that has a key size of 8 bytes; its key size is 56 bits long.

Configure an IKE Lifetime

The IKE lifetime sets the lifetime of an IKE SA. When the IKE SA expires, it is replaced by a new SA (and SPI) or terminated. The default value IKE lifetime is 3,600 seconds.

To configure IKE lifetime, include the lifetime-seconds statement and specify the number of seconds (180 through 86,400) at the [edit security ike proposal *ike-proposal-name*] hierarchy level:

```
[edit security ike proposal ike-proposal-name]  
lifetime-seconds seconds;
```

Example: Configure an IKE Proposal

Configure an IKE proposal:

```
[edit security ike]  
proposal ike-proposal {  
  authentication-method pre-shared-keys;  
  dh-group group1;  
  authentication-algorithm sha1;  
  encryption-algorithm 3des-cbc;  
}
```

Configure an IKE Policy for Preshared Keys

An IKE policy defines a combination of security parameters (IKE proposals) to be used during IKE negotiation. It defines a peer address, the preshared key for the given peer, and the proposals needed for that connection. During the IKE negotiation, IKE looks for an IKE policy that is the same on both peers. The peer that initiates the negotiation sends all its policies to the remote peer, and the remote peer tries to find a match.

A match is made when both policies from the two peers have a proposal that contains the same configured attributes. If the lifetimes are not identical, the shorter lifetime between the two policies (from the host and peer) is used. The configured preshared key must also match its peer.

You can create multiple, prioritized proposals at each peer to ensure that at least one proposal will match a remote peer's proposal.

First, you configure one or more IKE proposals; then you associate these proposals with an IKE policy. You can also prioritize a list of proposals used by IKE in the policy statement by listing the proposals you want to use, from first to last.

To configure an IKE policy, include the policy statement and specify a peer address at the [edit security ike] hierarchy level:

```
[edit security ike]
policy ike-peer-address [ ike-proposal ];
```



Note

The IKE policy peer address must be an IPSec tunnel destination address.

This section discusses the following topics:

Configure IKE Policy Mode on page 429

Configure IKE Policy Proposal on page 429

Configure IKE Policy Preshared Key on page 429

Configure IKE Policy Description on page 429

For an example of an IKE policy configuration, see “Example: Configure an IKE Policy” on page 430.

Configure IKE Policy Mode

IKE policy has two modes: aggressive and main. By default, main mode is enabled. Main mode uses six messages, in three exchanges, to establish the IKE SA. (These three steps are IKE SA negotiation, a Diffie-Hellman exchange, and authentication of the peer.) Main mode also allows a peer to hide its identity.

Aggressive mode also establishes an authenticated IKE SA and keys. However, aggressive mode uses half the number of messages, has less negotiation power, and does not provide identity protection. The peer can use the aggressive or main mode to start IKE negotiation; the remote peer accepts the mode sent by the peer.

To configure IKE policy mode, include the mode statement and specify aggressive or main at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike policy ike-peer-address]  
mode (aggressive | main);
```

Configure IKE Policy Proposal

The IKE policy proposal is a list of one or more proposals associated with an IKE policy.

To configure an IKE policy proposal, include the proposal statement and specify one or more proposal names at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike policy ike-peer-address]  
proposal [ ike-proposal-names ];
```

Configure IKE Policy Preshared Key

IKE policy preshared keys authenticate peers. You must manually configure a preshared key, which must match that of its peer. The preshared key can be an ASCII text (alphanumeric) key or a hexadecimal key.

To configure an IKE policy preshared key, include the pre-shared-key statement and a key at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike policy ike-peer-address]  
pre-shared-key (ascii-text key | hexadecimal key);
```

Configure IKE Policy Description

To specify a description for an IKE policy, include the description statement at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike policy ike-peer-address]  
description policy-description;
```

Example: Configure an IKE Policy

Define two IKE policies: policy 10.1.1.2 and policy 10.1.1.1. Each policy is associated with proposal-1 and proposal-2.

```
[edit security]
ike {
  proposal proposal-1 {
    authentication-method pre-shared-keys;
    dh-group group1;
    authentication-algorithm sha1;
    encryption-algorithm 3des-cbc;
    lifetime-seconds 1000;
  }
  proposal proposal-2 {
    authentication-method pre-shared-keys;
    dh-group group2;
    authentication-algorithm md5;
    encryption-algorithm des-cbc;
    lifetime-seconds 10000;
  }
  proposal proposal-3 {
    authentication-method rsa-signatures;
    dh-group group2;
    authentication-algorithm md5;
    encryption-algorithm des-cbc;
    lifetime-seconds 10000;
  }
  policy 10.1.1.2 {
    mode main;
    proposals [ proposal-1 proposal-2 ];
    pre-shared-key ascii-text example-pre-shared-key;
  }
  policy 10.1.1.1 {
    local-certificate certificate-file-name;
    local-key-pair private-public-key-file;
    mode aggressive;
    proposals [ proposal-2 proposal-3 ]
    pre-shared-key hexadecimal 0102030abbcdd;
  }
}
```

**Note**

Updates to the current IKE proposal and policy configuration are not applied to the current IKE SA; updates are applied to new IKE SAs.

If you want the new updates to take immediate effect, you must clear the existing IKE security associations so that they will be reestablished with the changed configuration. For information about how to clear the current IKE security association, see the *JUNOS Internet Software Operational Mode Command Reference: Protocols, Class of Service, Chassis, and Management*.

Configure an IPSec Proposal

An IPSec proposal lists protocols and algorithms (security services) to be negotiated with the remote IPSec peer.

To configure an IPSec proposal, include the proposal statement and specify an IPSec proposal name at [edit security ipsec] hierarchy level:

```
[edit security ipsec]
proposal ipsec-proposal-name;
authentication-algorithm (hmac-md5-96 | hmac-sha1-96);
encryption-algorithm (3des-cbc | des-cbc);
lifetime-seconds seconds;
protocol (ah | esp | bundle);
```

This section discusses the following topics:

Configure an Authentication Algorithm on page 431

Configure an Encryption Algorithm on page 432

Configure IPSec Lifetime on page 432

Configure the Protocol for the Dynamic SA on page 433

Configure an Authentication Algorithm

To configure an IPSec authentication algorithm, include the authentication-algorithm statement at the [edit security ipsec proposal *ipsec-proposal-name*] hierarchy level:

```
[edit security ipsec proposal ipsec-proposal-name]
authentication-algorithm (hmac-md5-96 | hmac-sha1-96);
```

The authentication algorithm can be one of the following:

hmac-md5-96—Hash algorithm that authenticates packet data. It produces a 128-bit digest. Only 96 bits are used for authentication.

hmac-sha1-96—Hash algorithm that authenticates packet data. It produces a 160-bit digest. Only 96 bits are used for authentication.

Configure an Encryption Algorithm

To configure an IPsec encryption algorithm, include the encryption-algorithm statement at the [edit security ipsec proposal *ipsec-proposal-name*] hierarchy level:

```
[edit security ipsec proposal ipsec-proposal-name]  
encryption-algorithm (3des-cbc | des-cbc);
```

The encryption algorithm can be one of the following:

3des-cbc—Encryption algorithm that has a block size of 24 bytes; its key size is 192 bits long.

des-cbc—Encryption algorithm that has a block size of 8 bytes; its key size is 48 bits long.



Note

We recommend that you use the 3DES-CBC encryption algorithm.

Configure IPsec Lifetime

The IPsec lifetime option sets the lifetime of an IPsec SA. When the IPsec SA expires, it is replaced by a new SA (and SPI) or terminated. If you do not configure a lifetime and a lifetime is not sent by a responder, the lifetime is 28,800 seconds.

To configure the IPsec lifetime, include the lifetime-seconds statement and specify the number of seconds (180 through 86,400) at the [edit security ipsec proposal *ipsec-proposal-name*] hierarchy level:

```
[edit security ipsec proposal ipsec-proposal-name]  
lifetime-seconds seconds;
```



Note

When a dynamic SA is created, two types of lifetimes are used: hard and soft. The hard lifetime specifies the lifetime of the SA. The soft lifetime, which is derived from the hard lifetime, informs the IPsec key management system that the SA is about to expire. This allows the key management system to negotiate a new SA before the hard lifetime expires.

Configure the Protocol for the Dynamic SA

The protocol statement sets the protocol for a dynamic SA. The ESP protocol can support authentication, encryption, or both. The AH protocol is used for strong authentication. AH also authenticates the IP packet. The bundle option uses AH authentication and ESP encryption, it does not use ESP authentication because AH provides stronger authentication of IP packets.

To configure the protocol for a dynamic SA, include the protocol statement and specify the ah, esp, or bundle option at the [edit security ipsec proposal *ipsec-proposal-name*] hierarchy level:

```
[edit security ipsec proposal ipsec-proposal-name]  
protocol (ah | esp | bundle);
```

Configure an IPsec Policy

An IPsec policy defines a combination of security parameters (IPsec proposals) used during IPsec negotiation. It defines Perfect Forward Secrecy (PFS) and the proposals needed for the connection. During the IPsec negotiation, IPsec looks for an IPsec proposal that is the same on both peers. The peer that initiates the negotiation sends all its policies to the remote peer, and the remote peer tries to find a match.

A match is made when both policies from the two peers have a proposal that contains the same configured attributes. If the lifetimes are not identical, the shorter lifetime between the two policies (from the host and peer) is used.

You can create multiple, prioritized IPsec proposals at each peer to ensure that at least one proposal will match a remote peer's proposal.

First, you configure one or more IPsec proposals first; then you associate these proposals with an IPsec policy. You can then prioritize a list of proposals used by IPsec in the policy statement by listing the proposals you want to use, from first to last.

To configure an IPsec policy, include the policy statement, specify the policy name and one or more proposals you want to associate with this policy at the [edit security ipsec] hierarchy level:

```
[edit security ipsec]  
policy ipsec-policy-name;
```

This section discusses the following topics related to configuring an IPsec policy:

Configure Perfect Forward Secrecy on page 434

Example: IPsec Policy Configuration on page 434

Configure Perfect Forward Secrecy

Perfect Forward Secrecy (PFS) provides additional security by means of a Diffie-Hellman shared secret value. With PFS, if one key is compromised, previous and subsequent keys are secure because they are not derived from previous keys. This statement is optional.

To configure PFS, include the perfect-forward-secrecy statement and specify a Diffie-Hellman group at the [edit security ipsec policy *ipsec-policy-name*] hierarchy level.

```
perfect-forward-secrecy {
  keys (group1 | group2);
}
```

The key can be one of the following:

group1—Specifies that IKE use the 768-bit Diffie-Hellman prime modulus group when performing the new Diffie-Hellman exchange.

group2—Specifies that IKE use the 1,024-bit Diffie-Hellman prime modulus group when performing the new Diffie-Hellman exchange.

group2 provides more security than group1, but requires more processing time.

Example: IPsec Policy Configuration

Defines an IPsec policy, dynamic policy-1, that is associated with two proposals (dynamic-1 and dynamic-2):

```
[edit security ipsec]
proposal dynamic-1 {
  protocol esp;
  authentication-algorithm hmac-md5-96;
  encryption-algorithm 3des-cbc;
  lifetime-seconds 6000;
}
proposal dynamic-2 {
  protocol esp;
  authentication-algorithm hmac-sha1-96;
  encryption-algorithm 3des-cbc;
  lifetime-seconds 6000;
}
policy dynamic-policy-1 {
  perfect-forward-secrecy {
    keys group1;
  }
  proposals [ dynamic-1 dynamic-2 ];
}
```

```
security-association dynamic-sa1 {  
  dynamic {  
    replay-window-size 64;  
    ipsec-policy dynamic-policy-1;  
  }  
}
```

**Note**

Updates to the current IPSec proposal and policy configuration are not applied to the current IPSec SA; updates are applied to new IPSec SAs.

If you want the new updates to take immediate effect, you must clear the existing IPSec security associations so that they will be reestablished with the changed configuration. For information about how to clear the current IPSec security association, see *JUNOS Internet Software Operational Mode Command Reference*.

Digital Certificates Guidelines

To define the digital certificate configuration, include the following statements at the [edit security certificates] and [edit security] hierarchy levels:

```
[edit security certificates]
certificates {
  cache-size bytes;
  cache-timeout-negative seconds;
  certification-authority ca-profile-name {
    ca-name ca-identity;
    encoding (binary | pem);
    crl file-name;
    file certificate-file-name;
    enrollment-url url-name;
    ldap-url url-name;
  }
  enrollment-retry number;
  local certificate-name;
  maximum-certificates number;
  path-length bytes;
}
```

```
[edit security ike]
policy ike-peer-address {
  description policy;
  encoding (binary | pem);
  identity identity-name;
  local-certificate certificate-file-name;
  local-key-pair private-public-key-file;
  mode (aggressive | main);
  proposal [ ike-proposal-names ];
}
```

For information about how to configure the description and mode statements, see “Configure IKE Policy Description” on page 429 and “Configure IKE Policy Mode” on page 429. For information about how to configure the IKE proposal, see “Configure IKE Policy Proposal” on page 429.



Note

For digital certificates, the JUNOS software supports only VeriSign.

To use digital certificates for dynamic SAs, perform the tasks described in the following sections:

1. Obtain a Certificate from a Certification Authority on page 437.
2. Generate a Private and Public Key on page 438
3. Configure Digital Certificates on page 438
4. Obtain a Publicly-Signed Router Digital Certificate on page 445.

Obtain a Certificate from a Certification Authority

Certificate authorities manage certificate requests and issue certificates to participating IPSec network devices. When you create a certificate request, you need to provide the information about the owner of the certificate. The required information and its format vary across certificate authorities.

Certificates use names in the X.500 format, a directory access protocol that provides both read and update access. The entire name is called a distinguished name (DN). It consists of a set of components, which often includes a common name (CN), an organization (O), an organization unit (OU), a country (C), a locality (L), and many others.



Note

For the dynamic registration of digital certificates, the JUNOS software supports only the simple certificate enrollment protocols (SCEP).

Issue the following command to obtain a public key certificate from a certification authority (CA) for your router. The results are saved in a specified file to the `/var/etc/ikecert` directory. The CA public key verifies certificates from remote routers.

```
user@host> request security certificate enroll filename filename ca-name
certificate-authority-name parameters parameters
```

Example: Obtain a Public Certificate from a Certification Authority

Specify a URL to the SCEP server and the name of the certification authority whose certificate you want: `mycompany.com`. `filename 1` is name of the file that stores the result. The output, "Received CA certificate:" provides the signature for the certificate, which allows you to verify (offline) that the certificate is genuine.

```
user@host> request security certificate enroll filename 1 ca-name mycompany.com
ca-file b url http://www.xyxcompany
URL: http://xyx.company
CA file: b
CA name: mycompany.com
Received CA certificate: 9b9651bb 290dc9e0 75c8030d 0d92606c
Saving certificate
```



Note

Each router is initially manually enrolled with a certification authority.

Generate a Private and Public Key

To generate a private and public key, issue the following command:

```
user@host> request security key-pair name size key-size type ( rsa | dsa )
```

Specifies the filename to store the public and private keys.

key-size can be 512, 1024, 1596, or 2048 bytes. The default key size is 1,024 bytes. *type* can be *rsa* or *dsa*. The default key is RSA.



Note

When you use SCEP, the JUNOS software only supports RSA.

Example: Generate Key Pair

Generate private and public key.

```
user@host> request security key-pair batt  
Generated key pair, key size 1024, file batt Algorithm RSA
```

Configure Digital Certificates

This section includes the following topics:

Configure the Certification Authority Properties on page 439

Configure the Cache Size on page 441

Configure the Negative Cache on page 441

Configure the Number of Enrollment Retries on page 442

Configure the Maximum Number of Peer Certificates on page 442

Configure the Path Length for the Certificate Hierarchy on page 442

Configure an IKE Policy for Digital Certificates on page 443

For information about the minimum digital certificate configuration for IKE, see “Minimum Digital Certificates Configuration for IKE” on page 415.

Configure the Certification Authority Properties

A certification authority is a trusted third-party organization that creates, enrolls, validates, and revokes digital certificates. The certification authority guarantees a user's identity and issues public and private “keys” for message encryption and decryption (coding and decoding).

To configure a certification authority, include the certification-authority statement at the [edit security certificates] hierarchy level:

```
[edit security certificates]
certification-authority ca-profile-name;
```

ca-profile-name is the CA profile name.

To configure certification authority properties, include statements at the [edit security certificates certification-authority *ca-profile-name*] hierarchy level:

```
[edit security certificates certification-authority ca-profile-name]
ca-name ca-identity;
crl file-name;
encoding (binary | pem);
enrollment-url url-name;
file certificate-file-name;
ldap-url url-name;
```

This section discusses the following topics:

Specify the Certification Authority Name on page 439

Configure the Certificate Revocation List on page 440

Configure the Type of Encoding Your CA Supports on page 440

Specify an Enrollment URL on page 440

Specify a File to Read the Digital Certificate on page 440

Specify an LDAP URL on page 441

Specify the Certification Authority Name

If you are enrolling with a CA using simple certificate enrollment protocols (SCEP), you need to specify the CA name (CA identity) that is used in the certificate request, in addition to the URL for the SCEP server.

To specify the name of the CA identity, include the *ca-name* statement at the [edit security certificates certification-authority *ca-profile-name*] hierarchy level:

```
[edit security certificates certification-authority ca-profile-name]
ca-name ca-identity;
```

ca-identity specifies the CA identity to use in the certificate request. It is typically the CA domain name.

Configure the Certificate Revocation List

A certificate revocation list (CRL) contains a list of digital certificates that have been cancelled before their expiration date. When a participating peer uses a digital certificate, it checks the certificate signature and validity. It also acquires the most recently issued CRL and checks that the certificate serial number is not on that CRL.

To configure the CA certificate revocation list, include the `crl` statement and specify the file to read the CRL from at the [edit security certificates certification-authority *ca-profile-name*] hierarchy level:

```
[edit security certificates certification-authority ca-profile-name]  
crl file-name;
```

Configure the Type of Encoding Your CA Supports

By default, encoding is set to binary. Encoding specifies the file format used for the local-certificate and local-key-pair statements. By default, the binary (distinguished encoding rules) format is enabled. PEM (Privacy Enhanced Mail) is an ASCII base 64 encoded format. Check with your CA to determine which file formats it supports.

To configure the file format that your CA supports, include the encoding statement and specify a binary or privacy enhanced email format (PEM) at the [edit security certificates certification-authority *ca-profile-name*] hierarchy level:

```
[edit security certificates certification-authority ca-profile-name]  
encoding ( binary | pem );
```

Specify an Enrollment URL

You specify the CA location, where your router should send SCEP -based certificate enrollment requests. To specify the CA location by naming the CA URL, include the enrollment-url statement at the [edit security certificates certification-authority *ca-profile-name*] hierarchy level:

```
[edit security certificates certification-authority ca-profile-name]  
enrollment-url url-name;
```

url-name is the CA location. The format is `http://CA_name`, where *CA_name* is the CA host DNS name or IP address.

Specify a File to Read the Digital Certificate

To specify which file to read the digital certificate from, include the file statement and specify the certificate file name at the [edit security certificates certification-authority *ca-profile-name*] hierarchy level:

```
[edit security certificates certification-authority ca-profile-name]  
file certificate-file-name;
```

Specify an LDAP URL

If your CA stores its current CRL at its LDAP (Lightweight Directory Access Protocol) server, you can optionally check your CA CRL list before using a digital certificate. If the digital certificate appears on the CA CRL, your router cannot use it. To access your CA CRL, include the `ldap-url` statement at the `[edit security certificates certification-authority ca-profile-name]` hierarchy level:

```
[edit security certificates certification-authority ca-profile-name]  
ldap-url url-name;
```

url-name is the certification authority LDAP server name. The format is `ldap://server_name`, where *server_name* is the CA host DNS name or IP address.

Configure the Cache Size

By default, cache size is 1 MB. To configure total cache size for digital certificates, include the `cache-size` statement at the `[edit security certificates]` hierarchy level:

```
[edit security certificates]  
cache-size bytes;
```

bytes is the cache size for digital certificates. The range can be 64 through 4,294,967,295 bytes.



Note

We recommend that you limit your cache size to 4 MB.

Configure the Negative Cache

Negative caching stores negative results and reduces the response time for negative answers. It also reduces the number of messages that are sent to the remote server. Maintaining a negative cache state allows the system to quickly return a failure condition when a lookup attempt is retried. Without a negative cache state, a retry would require waiting for the remote server to fail to respond, even though the system already “knows” that remote server is not responding.

By default, the negative cache is 60 seconds. To configure the negative cache, include the `cache-timeout-negative` statement at the `[edit security certificates]` hierarchy level:

```
[edit security certificates]  
cache-timeout-negative seconds;
```

seconds is the amount of time for which a failed CA or router certificate is present in the negative cache. While searching for certificates with a matching CA identity (domain name for certificates or CA domain name and serial for CRLs), the negative cache is searched first. If an entry is found in the negative cache, the search fails immediately.

**Note**

Configuring a large negative cache value can make you susceptible to a denial-of-service (DOS) attack.

Configure the Number of Enrollment Retries

By default, the number of enrollment retries is set to 0, an infinite number of retries. To specify how many times a router will resend a certificate request, include the enrollment-retry statement at the [edit security certificates] hierarchy level:

```
[edit security certificates]
enrollment-retry number;
```

number is the number of enrollment retries (0 through 100).

Configure the Maximum Number of Peer Certificates

By default, the maximum number of peer certificates to be cached is 1,024. To configure the maximum number of peer certificates to be cached, include the maximum-certificates statement at the [edit security certificates] hierarchy statement level:

```
[edit security certificates]
maximum-certificates number;
```

number is the maximum number of peer certificates to be cached. The range is 64 through 4,294,967,295 peer certificates.

Configure the Path Length for the Certificate Hierarchy

Certification authorities can issue certificates to other CAs. This creates a tree-like certification hierarchy. The highest trusted CA in the hierarchy is called the *trust anchor*. Sometimes the trust anchor is the root CA, which is usually signed by itself. In the hierarchy, every certificate is signed by the CA immediately above it. An exception is the root CA certificate, which is usually signed by the root CA itself. In general, a chain of multiple certificates may be needed, comprising a certificate of the public key owner (the end entity) signed by one CA, and zero or more additional certificates of CAs signed by other CAs. Such chains, called certification paths, are required because a public key user is only initialized with a limited number of assured CA public keys.

Path length refers to a path of certificates from one certificate to another certificate, based on the relationship of a CA and its “children”. When you configure the path length statement, you specify the maximum depth of the hierarchy to validate a certificate from the trusted root CA certificate to the certificate in question. For more information about the certificate hierarchy, see RFC 3280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.

By default, the maximum certificate path length is set to 15. The root anchor is 1. To configure path length, include the path-length statement at the [edit security certificates] hierarchy level:

```
[edit security certificates]
path-length certificate-path-length;
```

certificate-path-length is the maximum number certificates for the certificate path length. The range is 2 through 15 certificates.

Configure an IKE Policy for Digital Certificates

An IKE policy for digital certificates defines a combination of security parameters (IKE proposals) to be used during IKE negotiation. It defines a peer address and the proposals needed for that connection. During the IKE negotiation, IKE looks for an IKE policy that is the same on both peers. The peer that initiates the negotiation sends all its policies to the remote peer, and the remote peer tries to find a match.

To configure an IKE policy for digital certificates, include statements at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike]
policy ike-peer-address {
  encoding (binary | pem);
  identity identity-name;
  local-certificate certificate-file-name;
  local-key-pair private-public-key-file;
```

This section contains the following topics:

Configure the Type of Encoding Your CA Supports on page 443

Configure the Identity to Define the Remote Certificate Name on page 444

Specify the Certificate File Name on page 444

Specify the Private and Public Key File on page 444

Configure the Type of Encoding Your CA Supports

By default, the encoding is set to binary. Encoding specifies the file format used for the local-certificate and local-key-pair statements. By default, the binary (distinguished encoding rules) format is enabled. PEM (Privacy Enhanced Mail) is an ASCII base64 encoded format. Check with your CA to determine which file formats it supports.

To configure the file format that your CA supports, include the encoding statement and specify a binary or privacy enhanced email format (PEM) at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike policy-source ike-peer-address]
encoding (binary | pem);
```

Configure the Identity to Define the Remote Certificate Name

To define the remote certificate name, include the identity statement at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike policy ike-peer-address]  
identity identity-name;
```

identity-name defines the identity of the remote certificate name if the identity cannot be learned through IKE (ID payload or IP address).

Specify the Certificate File Name

To configure the certificate file name to read the local certificate from, include the local-certificate statement at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike policy ike-peer-address]  
local-certificate certificate-name;
```

certificate-file-name specifies the file to read the local certificate from.

Specify the Private and Public Key File

To specify the file name to read the public and private key from, include the local key-pair statement at the [edit security ike policy *ike-peer-address*] hierarchy level:

```
[edit security ike policy ike-peer-address]  
local-key-pair private-public-key-file;
```

private-public-key-file specifies the file to read the pair key from.

Obtain a Publicly-Signed Router Digital Certificate

To obtain a publicly-signed router digital certificate, issue the following command:

```
user@host> request security certificate enroll filename filename subject c=us,o=x
alternative-subject certificate-ip-address certification-authority certificate-authority key-file
key-file-name domain-name domain-name
```

Example: Obtain a Publicly-Signed Router Digital Certificate

Obtain a CA signed certificate by referencing the configured certification-authority statement "local". This statement is referenced by the request security certificate enroll filename m subject c=us,o=x alternative subject 1.1.1.1 certification-authority command.

```
[edit]
security {
  certificates {
    certification-authority local {
      ca-name test-ca1.company.com;
      file l;
      enrollment-url "http://www.xyzcompany.com";
    }
  }
}
```

To obtain a publicly-signed router digital certificate, issue the following command:

```
user@host> request security certificate enroll filename batt subject c=us,o=x
alternative-subject 1.1.1.1 certificate-authority local key-file y domain-name
abc.company.com
```

```
URL: http://www.xyz.company.com/scep/
CA file: l
local pub/private key pair: y
subject: c=us,o=x
domain-name abc.company.com
alternative subject: 1.1.1.1
Enrollment was successful. Certificate was written.
```

For information about how to use the operational mode commands to obtain a signed certificate, see the *JUNOS Internet Software Operational Mode Command Reference: Protocols, Class of Service, Chassis and Management*.

Another way to obtain a publicly-signed router digital certificate is to reference the configured statements such as the URL, CA name, and CA certificate file by means of the certification-authority statement:

```
user@host> request security certificate enroll filename m subject c=us,o=x
alternative-subject 1.1.1.1 certification-authority local key-file y domain-name
abc.company.com
```

Configure Trace Options

To configure security traceoptions, you can specify options in the traceoptions statement at the [edit security] hierarchy level:

```
[edit security]
traceoptions {
  file filename <files number> < size size>;
  flag all;
  flag database;
  flag general;
  flag ike;
  flag parse;
  flag policy-manager;
  flag routing-socket;
  flag timer;
}
```

The output of the security tracing options is placed in the /var/log/kmd.

You can specify one or more of the following security tracing flags:

- all—Trace all security events
- database—Trace database events
- general—Trace general events
- ike—Trace IKE module processing
- parse—Trace configuration processing
- policy-manager—Trace policy manager processing
- routing-socket—Trace routing socket messages
- timer—Trace internal timer events

Configure the ES PIC

Configuring the ES PIC associates the configured SA with a logical interface. This configuration defines the tunnel itself (logical subunit, tunnel addresses, maximum transmission unit [MTU], optional interface addresses, and the name of the SA to apply to traffic).

The addresses configured as the tunnel source and destination are the addresses in the outer IP header of the tunnel.



Note

The tunnel source address must be configured locally on the router, and the tunnel destination address must be a valid address for the security gateway terminating the tunnel.

The M5, M10, M20, and M40 routers support the ES PIC.

The SA must be a valid tunnel-mode SA. The interface address and destination address listed are optional. The destination address allows the user to configure a static route to encrypt traffic. If a static route uses that destination address as the next hop, traffic is forwarded through the portion of the tunnel in which encryption occurs. For more information about the ES PIC, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*.

Example: ES PIC Configuration

Configures an IPsec tunnel as a logical interface on the ES PIC. The logical interface specifies the tunnel through which the encrypted traffic travels. The ipsec-sa statement associates the security profile with the interface.

```
[edit interfaces]
es-0/0/0 {
  unit 0 {
    tunnel {
      source tunnel 10.5.5.5;           # tunnel source address
      destination 10.6.6.6;           # tunnel destination address
    }
    family inet {
      ipsec-sa ipsec-sa;               # name of security association to apply to
    }
  }
  packet
    address 10.1.1.8/32 {              # local interface address inside local VPN
      destination 10.2.2.254;         # destination address inside remote VPN
    }
  }
}
```

Configure Traffic

This section contains the following topics:

Traffic Overview on page 448

Example: Configure Outbound Traffic Filter on page 450

Example: Apply Outbound Traffic Filter on page 451

Example: Configure Inbound Traffic Filter for Policy Check on page 451

Example: Apply Inbound Traffic Filter to ES PIC for Policy Check on page 452

Traffic Overview

Traffic configuration defines the traffic that must flow through the tunnel. You configure outbound and inbound firewall filters, which identify and direct traffic to be encrypted and confirm that decrypted traffic parameters match those defined for the given tunnel. The outbound filter is applied to the LAN or WAN interface for the incoming traffic you want to encrypt off of that LAN or WAN. The inbound filter is applied to the ES PIC to check the policy for traffic coming in from the remote host. Because of the complexity of configuring a router to forward packets, no automatic checking is done to ensure the configuration is correct. Make sure that you configure the router very carefully.

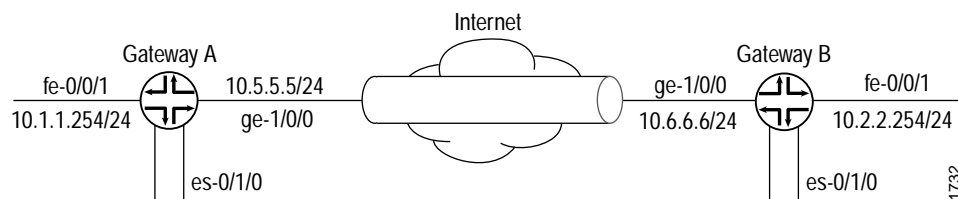


Note

The valid firewall filters statements for IPSec are destination-port, source-port, protocol, destination-address, and source-address.

In Figure 10, Gateway A protects the network 10.1.1.0/24, and Gateway B protects the network 10.2.2.0/24. The gateways are connected by an IPSec tunnel. For more information about firewalls, see the *JUNOS Internet Software Configuration Guide: Policy Framework*.

Figure 10: Example: IPSec Tunnel Connecting Security Gateways



The SA and ES interface for security Gateway A are configured as follows:

```
[edit security ipsec]
security-association manual-sa1 {
  manual {
    direction bidirectional {
      protocol esp;
      spi 2312;
      authentication {
        algorithm hmac-md5-96;
        key ascii-text 1234123412341234;
      }
      encryption {
        algorithm 3des-cbc;
        key ascii-text 123456789009876543211234;
      }
    }
  }
}

[edit interfaces es-0/1/0]
unit 0 {
  tunnel {
    source 10.5.5.5;
    destination 10.6.6.6;
  }
  family inet {
    ipsec-sa manual-sa1;
    address 10.1.1.8/32 {
      destination 10.2.2.254;
    }
  }
}
```

Example: Configure Outbound Traffic Filter

Firewall filters for outbound traffic direct the traffic through the desired IPsec tunnel and ensure that the tunneled traffic goes out the appropriate interface (see Figure 10). Here, an outbound firewall filter is created on security Gateway A; it identifies the traffic to be encrypted and adds it to the input side of the interface that carries the internal VPN traffic:

```
[edit firewall]
filter ipsec-encrypt-policy-filter {
  term term1 {
    from {
      source-address {      # local network
        10.1.1.0/24;
      }
      destination-address { # remote network
        10.2.2.0/24;
      }
    }
    then ipsec-sa manual-sa1; # apply SA name to packet
  }
  term default {
    then accept;
  }
}
```



Note

The source address, port, and protocol on the outbound traffic filter must match the destination address, port, and protocol on the inbound traffic filter. The destination address, port, and protocol on the outbound traffic filter must match the source address, port, and protocol on the inbound traffic filter.

Example: Apply Outbound Traffic Filter

After you've configured the outbound firewall filter, you apply it:

```
[edit interfaces]
fe-0/0/1 {
  unit 0 {
    family inet {
      filter {
        input ipsec-encrypt-policy-filter;
      }
      address 10.1.1.254/24;
    }
  }
}
```

The outbound filter is applied on the Fast Ethernet interface at the [edit interfaces fe-0/0/1 unit 0 family inet] hierarchy level. Any packet matching the IPSec action term (term 1) on the input filter (ipsec-encrypt-policy-filter), configured on the Fast Ethernet interface, is directed to the ES PIC interface at the [edit interfaces es-0/1/0 unit 0 family inet] hierarchy level. So, if a packet arrives from the source address 10.1.1.0/24 and goes to the destination address 10.2.2.0/24, the Packet Forwarding Engine directs the packet to the ES PIC interface, which is configured with the manual-sa1 SA. The ES PIC receives the packet, applies the manual-sa1 SA, and sends the packet through the tunnel.

The router must have a route to the tunnel endpoint; add a static route if necessary.

Example: Configure Inbound Traffic Filter for Policy Check

Here, an inbound firewall filter, which performs the final IPSec policy check, is created on security gateway A. This check ensures that only packets that match the traffic configured for this tunnel are accepted.

```
filter ipsec-decrypt-policy-filter {
  term term1 {
    from {
      source-address {
        10.2.2.0/24;
      }
      destination-address {
        10.1.1.0/24;
      }
    }
  }
  then accept;
}
```

Example: Apply Inbound Traffic Filter to ES PIC for Policy Check

After you create the inbound firewall filter, apply it to the ES PIC. Here, the inbound firewall filter (ipsec-decrypt-policy-filter) is applied on the decrypted packet to perform the final policy check. The IPsec manual-sa1 SA is referenced at the [edit interfaces es-1/2/0 unit 0 family inet] hierarchy level and decrypts the incoming packet.

```
[edit interfaces]
es-1/2/0 {
  unit 0 {
    tunnel {
      source 10.5.5.5;           # tunnel source address
      destination 10.6.6.6;     # tunnel destination address
    }
    family inet {
      filter {
        input ipsec-decrypt-policy-filter;
      }
      ipsec-sa manual-sa1;      # SA name applied to packet
      address 10.1.1.8/32 {     # local interface address inside local VPN
        destination 10.2.2.254; # destination address inside remote VPN
      }
    }
  }
}
```

The Packet Forwarding Engine directs IPsec packets to the ES PIC. It uses the packet's security parameter index (SPI), protocol, and destination address to look up the SA configured on one of the ES interfaces. The IPsec manual-sa1 SA is referenced at the [edit interfaces es-1/2/0 unit 0 family inet] hierarchy level and is used to decrypt the incoming packet. When the packets are processed (decrypted, authenticated, or both), the input firewall filter (ipsec-decrypt-policy-filter) is applied on the decrypted packet to perform the final policy check. Term1 defines the decrypted (and verified) traffic and performs the required policy check.



Note

The inbound traffic filter is applied after the ES PIC has processed the packet, so the decrypted traffic is defined as any traffic that the remote gateway is encrypting and sending to this router. IKE uses this filter to determine the policy required for a tunnel. This policy is used during the negotiation with the remote gateway to find the matching SA configuration.

Configure an ES Tunnel Interface for a Layer 3 VPN

To configure an ES tunnel interface for a Layer 3 VPN, you need to configure an ES tunnel interface on the PE router and on the CE router. You also need to configure Internet Protocol Security (IPSec) on the PE and CE routers. For more information about configuring an ES tunnel for a Layer 3 VPN, see the *JUNOS Internet Software Configuration Guide: VPNs*.

JUNOScript XNM-SSL Service

This section contains the following topics:

Configure JUNOScript XNM-SSL Service on page 453

Load the SSL Certificate from A File or URL on page 454

Configure JUNOScript XNM-SSL Service

The Secure Sockets Layer (SSL) protocol uses public-private key technology, which requires a paired private key and authentication certificate xnm-ssl service. This section describes how to import the SSL certificate into the JUNOScript server machine. For a complete example on how to configure the xnm-ssl service, see the *JUNOScript API Guide*.



Note

Configuring xnm-ssl service does not apply to IPSec.

To import an SSL certificate into the router, include the local statement at the [edit security certificates] hierarchy level:

```
[edit security certificates]
local certificate-name;
```

Enter CLI configuration mode on the JUNOScript server machine and issue the following commands at the [edit security certificates] and [edit security certificates local *certificate-name*] hierarchy levels to import the certificate.

```
[edit security certificates]
user@host# edit local certificate-name
```

where *certificate-name* is the name of the certificate.

```
[edit security certificates local certificate-name]
user@host# set load-key-file URL_or_path
```

where *URL_or_Path* is the URL or pathname on the local disk.



Note

The CLI expects the private key in the specified file (*URL_or_path*) to be unencrypted. If the key is encrypted, the CLI prompts for the passphrase associated with it, decrypts it, and stores the unencrypted version.

Load the SSL Certificate from A File or URL

To load a SSL certificate from a file or URL, include the local statement at the [edit security certificates] hierarchy level:

```
[edit security]
certificates {
  local local-certificate-name {
    load-key-file (file-name | url);
  }
}
```

local-certificate-name is the name of the SSL certificate name that want to load.

file-name is the name of the file that contains an SSL certificate and private key in PEM format

url is the SSL certificate and private key (in Privacy Enhanced Mail format) location.