

## Chapter 32

# Security Services Configuration Guidelines

To configure security services, include the following 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;
    crl file-name;
    encoding (binary | pem);
    enrollment-url url-name;
    file certificate-filename;
    ldap-url url-name;
  }
  enrollment-retry attempts;
  local certificate-filename {
    certificate-key-string;
    load-key-file key-filename;
  }
  maximum-certificates number;
  path-length certificate-path-length;
}
ike {
  proposal ike-proposal-name {
    authentication-algorithm (md5 | sha1);
    authentication-method (dsa-signatures | pre-shared-keys | rsa-signatures);
    description description;
    dh-group (group1 | group2);
    encryption-algorithm (3des-cbc | des-cbc);
    lifetime-seconds seconds;
  }
}
```

```

policy ike-peer-address {
  description description;
  encoding (binary | pem);
  identity identity-name;
  local-certificate certificate-filename;
  local-key-pair private-public-key-file;
  mode (aggressive | main);
  pre-shared-key (ascii-text key | hexadecimal key);
  proposals [ proposal-names ];
}
}
ipsec {
  proposal ipsec-proposal-name {
    authentication-algorithm (hmac-md5-96 | hmac-sha1-96);
    description description;
    encryption-algorithm (3des-cbc | des-cbc);
    lifetime-seconds seconds;
    protocol (ah | esp | bundle);
  }
  policy ipsec-policy-name {
    description description;
    perfect-forward-secrecy {
      keys (group1 | group2);
    }
    proposals [ proposal-names ];
  }
  security-association sa-name {
    description description;
    dynamic {
      ipsec-policy policy-name;
      replay-window-size (32 | 64);
    }
    manual {
      direction (inbound | outbound | bi-directional) {
        authentication {
          algorithm (hmac-md5-96 | hmac-sha1-96);
          key (ascii-text key | hexadecimal key);
        }
        auxiliary-spi auxiliary-spi;
        encryption {
          algorithm (des-cbc | 3des-cbc);
          key (ascii-text key | hexadecimal key);
        }
        protocol (ah | esp | bundle);
        spi spi-value;
      }
    }
    mode (tunnel | transport);
  }
}
}

```

```

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 IP Security (IPSec) monitoring and troubleshooting, see the *JUNOS Protocols, Class of Service, and System Basics Command Reference*.

---

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

Minimum Manual SA Configuration on page 624

Minimum IKE Configuration on page 624

Minimum Digital Certificates Configuration for IKE on page 625

Configuring Security Associations on page 625

Configuring an IKE Proposal (Dynamic SAs Only) on page 634

Configuring an IKE Policy for Preshared Keys on page 637

Configuring an IPSec Proposal on page 640

Configuring the IPSec Policy on page 643

Configuring Digital Certificates on page 645

Configuring Trace Options on page 656

Configuring the ES PIC on page 656

Configuring Traffic on page 657

Configuring an ES Tunnel Interface for a Layer 3 VPN on page 662

Using JUNOScript SSL Service on page 663

## Minimum Manual SA Configuration

---

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

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

## Minimum IKE Configuration

---

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

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

## Minimum Digital Certificates Configuration for IKE

---

To define a digital certificates configuration for IKE, include at least the following statements at the [edit security certificates] and [edit security ike] hierarchy levels:

```
[edit security]
certificates {
  certification-authority ca-profile-name {
    ca-name ca-identity;
    crl file-name;
    enrollment-url url-name;
    file certificate-filename;
    ldap-url url-name;
  }
}
ike {
  policy ike-peer-address {
    local-certificate certificate-filename;
    local-key-pair private-public-key-file;
    proposal [ ike-proposal-names ];
  }
  proposal ike-proposal-name {
    authentication-method rsa-signatures;
  }
}
```

## Configuring Security Associations

---

To use IPSec security services, you create an 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 “Configuring Manual Security Associations” on page 628.

**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 “Configuring the ES PIC” on page 656.



**NOTE:** The JUNOS software does not perform a commit check when an SA name referenced in the Border Gateway Protocol (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 Physical Interface Card (PIC).

---

To configure an SA for IPSec, include the security-association statement at the [edit security ipsec] hierarchy level:

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

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

Configuring the Description for an SA on page 626

Configuring IPSec Mode on page 626

Configuring Manual Security Associations on page 628

Configuring Dynamic Security Associations on page 633

### **Configuring the Description for an SA**

To specify a description for an IPSec SA, include the description statement at the [edit security ipsec security-association *sa-name*] hierarchy level:

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

### **Configuring 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:

Configuring Transport Mode on page 626

Configuring Tunnel Mode on page 627

#### **Configuring 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. Virtual private network (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 with the transport option at the [edit security ipsec security-association *sa-name*] hierarchy level:

```
[edit security ipsec security-association sa-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 Routing Protocols Configuration Guide*.



**NOTE:** You can configure BGP to peer over encrypted tunnels.

---

### Configuring Tunnel Mode

You use tunnel mode when you use preshared keys with IKE to authenticate peers, or digital certificates with IKE to authenticate peers. In tunnel mode, encryption services are performed on an ES PIC.

When 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 certificate 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.

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

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



**NOTE:** Tunnel mode requires the ES PIC.

The JUNOS software supports only BGP in transport mode.

---

To enable tunnel mode, follow the steps in these sections:

1. Configuring an IKE Proposal (Dynamic SAs Only) on page 634
2. Configuring Security Associations on page 625
3. Configuring the ES PIC on page 656
4. Configuring Traffic on page 657

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

## Configuring Manual Security Associations

Manual SAs require no negotiation; all values, including the keys, are static and specified in the configuration. As a result, peers can communicate only when they all share the same configured options.

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

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

The following sections describe how to configure a manual SA:

Configuring the Processing Direction on page 628

Configuring the Protocol for a Manual SA on page 629

Configuring the Security Parameter Index on page 630

Configuring the Auxiliary Security Parameter Index on page 630

Configuring the Authentication Algorithm and Key on page 631

Configuring the Encryption Algorithm and Key on page 632

### Configuring the Processing 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 *sa-name* manual] hierarchy level:

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

For sample configurations, see the following sections:

Example: Configuring Inbound and Outbound Processing on page 629

Example: Configuring Bidirectional Processing on page 629

### **Example: Configuring Inbound and Outbound Processing**

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

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

### **Example: Configuring Bidirectional Processing**

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

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

### **Configuring the Protocol for a Manual SA**

IPSec uses two protocols to protect IP traffic: Encapsulating Security Payload (ESP) and authentication header (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.



**NOTE:** The AH protocol is supported only on M-series platforms.

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

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

### Configuring the Security Parameter Index

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. For more information, see “Configuring the Auxiliary Security Parameter Index” on page 630.

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

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

### Configuring the Auxiliary Security Parameter Index

When you configure the protocol statement to use the bundle option, the JUNOS software uses the auxiliary SPI for the ESP and the SPI for the AH.



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

To configure the auxiliary SPI, include the auxiliary-spi statement at the [edit security ipsec security-association *sa-name* manual direction (inbound | outbound | bi-directional)] hierarchy level and set the value to an integer between 256 and 16,639:

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

### Configuring the Authentication Algorithm and Key

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

```
[edit security ipsec security-association sa-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 *key***—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 *key***—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.

### Configuring the Encryption Algorithm and Key

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

```
[edit security ipsec security-association sa-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 Data Encryption Standard (DES) 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 that 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 the AH protocol.

---

## Configuring 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 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 “Configuring an IKE Policy for Preshared Keys” on page 637 and “Configuring an IKE Proposal (Dynamic SAs Only)” on page 634. For more information about IPSec policies and proposals, see “Configuring the IPSec Policy” on page 643.



**NOTE:** Dynamic tunnel SAs require an ES PIC.

---

To configure a dynamic SA, include the dynamic statement at the [edit security ipsec security-association *sa-name*] hierarchy level. Specify an IPSec policy name and optionally a 32- or 64-packet replay window size.

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



**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.

---

## Configuring an IKE Proposal (Dynamic SAs Only)

---

Dynamic SAs require IKE configuration. 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 and define its properties, include the following statements at the [edit security ike] hierarchy level:

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

For information about associating an IKE proposal with an IKE policy, see “Associating Proposals with an IKE Policy” on page 638.

This section discusses the following topics:

Configuring the Authentication Algorithm for an IKE Proposal on page 634

Configuring the Authentication Method for an IKE Proposal on page 635

Configuring the Description for an IKE Proposal on page 635

Configuring the Diffie-Hellman Group for an IKE Proposal on page 635

Configuring the Encryption Algorithm for an IKE Proposal on page 636

Configuring the Lifetime for an IKE SA on page 636

Example: Configuring an IKE Proposal on page 636

### **Configuring the Authentication Algorithm for an IKE Proposal**

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.

### **Configuring the Authentication Method for an IKE Proposal**

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 (DSA)

pre-shared-keys—Preshared keys; a key derived from an out-of-band mechanism is used to authenticate an exchange

rsa-signatures—A public key algorithm, which supports encryption and digital signatures

### **Configuring the Description for an IKE Proposal**

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

```
[edit security ike proposal ike-proposal-name]
description description;
```

### **Configuring the Diffie-Hellman Group for an IKE Proposal**

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 1024-bit Diffie-Hellman prime modulus group when performing the new Diffie-Hellman exchange.

group2 provides more security but requires more processing time.

### **Configuring the Encryption Algorithm for an IKE Proposal**

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.

### **Configuring the Lifetime for an IKE SA**

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 is terminated. The default value IKE lifetime is 3600 seconds.

To configure the 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: Configuring 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;  
}
```

## Configuring 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 at the [edit security ike] hierarchy level and specify a peer address:

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



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

---

This section discusses the following topics:

Configuring the Description for an IKE Policy on page 638

Configuring the Mode for an IKE Policy on page 638

Configuring the Preshared Key for an IKE Policy on page 638

Associating Proposals with an IKE Policy on page 638

Example: Configuring an IKE Policy on page 639

### **Configuring the Description for an IKE Policy**

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 description;
```

### **Configuring the Mode for an IKE Policy**

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);
```

### **Configuring the Preshared Key for an IKE Policy**

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 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);
```

### **Associating Proposals with an IKE Policy**

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 proposals statement at the [edit security ike policy *ike-peer-address*] hierarchy level and specify one or more proposal names:

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

For more information about configuring an individual proposal, see “Configuring an IKE Proposal (Dynamic SAs Only)” on page 634.

**Example: Configuring 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-filename;
    local-key-pair private-public-key-file;
    mode aggressive;
    proposals [ proposal-2 proposal-3 ]
    pre-shared-key hexadecimal 0102030abbcd;
  }
}
```



**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 Protocols, Class of Service, and System Basics Command Reference*.

---

## Configuring 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 and define its properties, include the following statements at the [edit security ipsec] hierarchy level:

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

This section discusses the following topics:

Configuring the Authentication Algorithm for an IPsec Proposal on page 641

Configuring the Description for an IPsec Proposal on page 641

Configuring the Encryption Algorithm for an IPsec Proposal on page 641

Configuring the Lifetime for an IPsec SA on page 641

Configuring the Protocol for a Dynamic IPsec SA on page 643

### Configuring the Authentication Algorithm for an IPSec Proposal

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.

### Configuring the Description for an IPSec Proposal

To specify a description for an IPSec proposal, include the description statement at the [edit security ipsec proposal *ipsec-proposal-name*] hierarchy level:

```
[edit security ike policy ipsec-proposal-name]  
description description;
```

### Configuring the Encryption Algorithm for an IPSec Proposal

To configure the 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 triple DES cipher block chaining (3DES-CBC) encryption algorithm.

---

### Configuring the Lifetime for an IPSec SA

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 is terminated. A new SA has new authentication and encryption keys, and SPI; however, the algorithms may remain the same if the proposal is not changed. 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. When you specify the lifetime, you specify a hard lifetime.

---

## Configuring the Protocol for a Dynamic IPsec 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 at the [edit security ipsec proposal *ipsec-proposal-name*] hierarchy level:

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

## Configuring the 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; then you associate these proposals with an IPsec policy. You can prioritize the proposals in the list by listing them in the order in which the IPsec policy uses them (first to last).

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

```
[edit security ipsec]
policy ipsec-policy-name {
    proposals [ proposal-names ];
}
```

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

Configuring Perfect Forward Secrecy on page 644

Example: IPsec Policy Configuration on page 644

### Configuring 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:

```
[edit security ipsec policy ipsec-policy-name]
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 1024-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

Define 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 the *JUNOS Protocols, Class of Service, and System Basics Command Reference*.

## Configuring Digital Certificates

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

```

[edit security]
certificates {
  cache-size bytes;
  cache-timeout-negative seconds;
  certification-authority ca-profile-name {
    ca-name ca-identity;
    crl file-name;
    encoding (binary | pem);
    enrollment-url url-name;
    file certificate-filename;
    ldap-url url-name;
  }
  enrollment-retry attempts;
  local certificate-filename {
    certificate-key-string;
    load-key-file key-filename;
  }
  maximum-certificates number;
  path-length certificate-path-length;
}
ike {
  policy ike-peer-address {
    description policy;
    encoding (binary | pem);
    identity identity-name;
    local-certificate certificate-filename;
    local-key-pair private-public-key-file;
    mode (aggressive | main);
    pre-shared-key (ascii-text key | hexadecimal key);
    proposals [ proposal-names ];
  }
}

```

For information about how to configure the description and mode statements, see “Configuring the Description for an IKE Policy” on page 638 and “Configuring the Mode for an IKE Policy” on page 638. For information about how to configure the IKE proposal, see “Associating Proposals with an IKE Policy” on page 638.



**NOTE:** For digital certificates, the JUNOS software supports only VeriSign.

---

## Overview

Digital certificates provide a way of authenticating users through a trusted third party called a certificate authority (CA). The CA validates the identity of a certificate holder and "signs" the certificate to attest that it has not been forged or altered.

A certificate includes the following information:

- The distinguished name (DN) of the owner. A DN is a unique identifier and consists of a fully qualified name including not only the common name (CN) of the owner, the owner's organization, and other distinguishing information.

- The public key of the owner.

- The date on which the certificate was issued.

- The date on which the certificate expires.

- The distinguished name of the issuing CA.

- The digital signature of the issuing CA.

The additional information in a certificate allows recipients to decide whether to accept the certificate. The recipient can determine if the certificate is still valid based on the expiration date. The recipient can check whether the CA is trusted by the site based on the issuing CA.

With a certificate, a CA takes the owner's public key, signs that public key with the its own private key, and returns this to the owner as a certificate. The recipient can extract the certificate (containing the CA's signature) with the owner's public key. By using the CA's public key and the CA's signature on the extracted certificate, the recipient can validate the CA's signature and owner of the certificate.

When you use digital certificates, your first step is to send in a request to obtain a certificate from your CA. You then configure digital certificates and a digital certificate IKE policy. Finally, you obtain a digitally signed certificate from a CA.

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

1. Obtaining a Certificate from a Certificate Authority on page 647
2. Configuring Digital Certificates on page 648
3. Configuring an IKE Policy for Digital Certificates on page 653
4. Obtaining a Signed Certificate from the CA on page 654

### Obtaining a Certificate from a Certificate 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 DN (distinguished name). It consists of a set of components, which often includes a CN (common name), an organization (O), an organization unit (OU), a country (C), a locality (L), and so on.



**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 CA. The results are saved in the specified file in the `/var/etc/ikecert` directory. The CA public key verifies certificates from remote peers.

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

For more information, see the following sections:

Example: Obtaining a Certificate from a Certificate Authority on page 647

Generating a Private and Public Key on page 648

#### Example: Obtaining a Certificate from a Certificate 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 ca_verisign ca-file
verisign ca-name xyzcompany url
http://pilotsiteipsec.verisign.com/cgi-bin/pkiclient.exe
URL: http://pilotsiteipsec.verisign.com/cgi-bin/pkiclient.exe CA name:
juniper.net CA file: verisign Encoding: binary
Certificate enrollment has started. To see the certificate enrollment status, check
the key management process (kmd) log file at /var/log/kmd. <-----
```



**NOTE:** Each router is initially manually enrolled with a certificate authority.

---

### Generating 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 )
```

*name* specifies the filename in which to store the public and private keys.

*key-size* can be 512, 1024, 1596, or 2048 bytes. The default key size is 1024 bytes.

*type* can be rsa or dsa. The default is RSA.



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

---

### Example: Generating a Key Pair

Generate a private and public key:

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

## Configuring Digital Certificates

This section includes the following topics:

Configuring the Certificate Authority Properties on page 649

Configuring the Cache Size on page 651

Configuring the Negative Cache on page 651

Configuring the Number of Enrollment Retries on page 652

Configuring the Maximum Number of Peer Certificates on page 652

Configuring the Path Length for the Certificate Hierarchy on page 652

For information about the minimum digital certificate configuration for IKE, see “Minimum Digital Certificates Configuration for IKE” on page 625.

## Configuring the Certificate Authority Properties

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

To configure a certificate authority and its properties, include the following statements at the [edit security certificates] 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-filename;
  ldap-url url-name;
```

*ca-profile-name* is the CA profile name.

This section discusses the following topics:

- Specifying the Certificate Authority Name on page 649
- Configuring the Certificate Revocation List on page 650
- Configuring the Type of Encoding Your CA Supports on page 650
- Specifying an Enrollment URL on page 650
- Specifying a File to Read the Digital Certificate on page 650
- Specifying an LDAP URL on page 651

### **Specifying the Certificate 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.

**Configuring 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 from which to read the CRL at the [edit security certificates certification-authority *ca-profile-name*] hierarchy level:

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

**Configuring 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. Privacy-enhanced mail (PEM) 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 PEM format at the [edit security certificates certification-authority *ca-profile-name*] hierarchy level:

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

**Specifying 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.

**Specifying a File to Read the Digital Certificate**

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

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

### Specifying an LDAP URL

If your CA stores its current CRL at its Lightweight Directory Access Protocol (LDAP) 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.

### Configuring the Cache Size

By default, the cache size is 2 megabytes (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 from 64 through 4,294,967,295 bytes.



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

---

### Configuring 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 20 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.

---

### Configuring 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 attempts;
```

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

### Configuring the Maximum Number of Peer Certificates

By default, the maximum number of peer certificates to be cached is 1024. 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 from 64 through 4,294,967,295 peer certificates.

### Configuring 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 from 2 through 15 certificates.

## Configuring 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 the following 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-filename;
  local-key-pair private-public-key-file;
}
```

This section contains the following topics:

Configuring the Type of Encoding Your CA Supports on page 653

Configuring the Identity to Define the Remote Certificate Name on page 653

Specifying the Certificate Filename on page 654

Specifying the Private and Public Key File on page 654

### Configuring 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 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 PEM format at the [edit security ike policy *ike-peer-address*] hierarchy level:

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

### Configuring 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).

### Specifying the Certificate Filename

To configure the certificate filename from which to read the local certificate, 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-filename;
```

*certificate-filename* specifies the file from which to read the local certificate.

### Specifying the Private and Public Key File

To specify the filename from which to read the public and private key, 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 from which to read the pair key.

## Obtaining a Signed Certificate from the CA

To obtain signed certificate from the CA, 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
```

The results are saved in a specified file to the /var/etc/ikecert directory.

### Example: Obtaining a Signed 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 xyz.company.com;
      file l;
      enrollment-url "http://www.xyzcompany.com";
    }
  }
}
```

To obtain a signed certificate from the CA, issue the following command:

```
user@host> request security certificate enroll filename l subject c=uk,o=london
alternative-subject 10.50.1.4 certification-authority verisign key-file host-1.prv
domain-name host.xyzcompany.com
CA name: xyz.company.com CA file: ca_verisign
local pub/private key pair: host.prv
subject: c=uk,o=london domain name: host.juniper.net
alternative subject: 10.50.1.4
Encoding: binary
Certificate enrollment has started. To see the certificate enrollment status, check
the key management process (kmd) log file at /var/log/kmd. <-----
```

For information about how to use the operational mode commands to obtain a signed certificate, see the *JUNOS Protocols, Class of Service, and System Basics Command Reference*.

Another way to obtain a signed certificate from the CA 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
```

## Configuring 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 file.

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

## Configuring 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.

You can also configure IPsec on the Adaptive Services PIC. For information about how to configure IPsec on a Adaptive Services PIC, see the *JUNOS Services Interfaces Configuration Guide*.

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 Services Interfaces Configuration Guide*.

### Example: Configuring the ES PIC

Configure 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
      }
    }
  }
}
```

## Configuring Traffic

This section contains the following topics:

Traffic Overview on page 658

Example: Configuring Outbound Traffic Filter on page 660

Example: Applying Outbound Traffic Filter on page 661

Example: Configuring Inbound Traffic Filter for Policy Check on page 661

Example: Applying Inbound Traffic Filter to ES PIC for Policy Check on page 662

## Traffic Overview

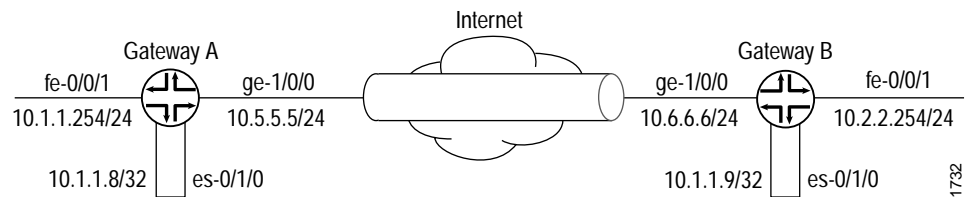
Traffic configuration defines the traffic that must flow through the tunnel. You configure outbound and inbound firewall filters, which identify and direct traffic to be encrypted and confirm that decrypted traffic parameters match those defined for the given tunnel. The outbound filter is applied to 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 that 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 11, 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 Policy Framework Configuration Guide*.

**Figure 11: 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 10.1.1.9;
    }
  }
}
}
```

The SA and ES interface for security Gateway B 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 10.6.6.6;
    destination 10.5.5.5;
  }
  family inet {
    ipsec-sa manual-sa1;
    address 10.1.1.9/32 {
      destination 10.1.1.8;
    }
  }
}
}
```

**Example: Configuring 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 11). 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: Applying Outbound Traffic Filter**

After you have 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. 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: Configuring 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: Applying 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 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.

### Configuring 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 provider edge (PE) router and on the customer edge (CE) router. You also need to configure IPsec on the PE and CE routers. For more information about configuring an ES tunnel for a Layer 3 VPN, see the *JUNOS VPNs Configuration Guide*.

## Using JUNOScript SSL Service

---

This section contains the following topics:

Configuring the JUNOScript SSL Service on page 663

Loading the SSL Certificate from a File or URL on page 664

### Configuring the JUNOScript SSL Service

The Secure Sockets Layer (SSL) protocol uses public-private key technology, which requires a paired private key and authentication certificate SSL service. This section describes how to import the SSL certificate into the JUNOScript server. 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-filename;
```

To import the certificate, 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 level :

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

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

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

*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.

---

### ***Loading the SSL Certificate from a File or URL***

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

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

*local-certificate-filename* is the name of the SSL certificate name that you 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 PEM location.