

Public Key Infrastructure User Guide

Published
2024-12-18

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Public Key Infrastructure User Guide

Copyright © 2024 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

1

[About This Guide | vii](#)

Public Key Infrastructure

[What is PKI | 2](#)

[PKI Overview | 2](#)

[PKI Fundamentals | 5](#)

[PKI in Junos OS | 10](#)

[PKI Components in Junos OS | 12](#)

PKI Certificates | 15

[Digital Certificates | 15](#)

[| Generate Digital Certificates Manually: Configuration Overview | 16](#)

[Certificate Authority | 16](#)

[Configure a Trusted CA Group | 17](#)

[| Create a Trusted CA Group for a List of CA Profiles | 17](#)

[Delete a CA Profile from a Trusted CA Group | 18](#)

[Delete a Trusted CA Group | 19](#)

[Certificate Authority Profiles | 20](#)

[Example: Configure a CA Profile | 21](#)

[| Requirements | 21](#)

[Overview | 21](#)

[Configuration | 22](#)

[Verification | 23](#)

[Example: Configure an IPv6 address as the Source Address for a CA Profile | 23](#)

[Self-Signed Digital Certificates | 24](#)

[Self-Signed Certificates | 25](#)

[Example: Generate a Public-Private Key Pair | 26](#)

[| Requirements | 26](#)

[Overview | 26](#)

[Configuration | 26](#)

- Verification | 27

Example: Manually Generate Self-Signed Certificates | 27

- Requirements | 27

- Overview | 27

- Configuration | 27

- Verification | 28

Using Automatically Generated Self-Signed Certificates (CLI Procedure) | 28

Manage PKI Certificates | 30

Enroll Certificate | 30

Enroll Digital Certificates Online: Configuration Overview | 31

Online CA Certificate Enrollment | 31

Local Certificate Requests | 31

Enroll a CA Certificate Online Using SCEP | 32

Example: Enroll a Local Certificate Online Using SCEP | 33

- Requirements | 33

- Overview | 33

- Configuration | 34

- Verification | 35

Example: Using SCEP to Automatically Renew a Local Certificate | 35

- Requirements | 35

- Overview | 36

- Configuration | 36

- Verification | 37

CMPv2 and SCEP Certificate Enrollment | 37

Certificate Enrollment with CMPv2 | 38

Example: Manually Generate a CSR for the Local Certificate and Send it to the CA Server | 40

- Requirements | 40

- Overview | 40

- Configuration | 41

- Verification | 41

Example: Load CA and Local Certificates Manually | 42

- Requirements | 42

- Overview | 42

- Configuration | 43

- Verification | 43

Delete Certificates (CLI Procedure) | 44

Certificate Revocation | 44

Online Certificate Status Protocol and Certificate Revocation Lists | 45

Example: Manually Load a CRL onto the Device | 48

Requirements | 48

Overview | 48

Configuration | 49

Verification | 49

Dynamic CRL Download and Verify | 49

Example: Configure a Certificate Authority Profile with CRL Locations | 52

Requirements | 52

Overview | 53

Configuration | 53

Verification | 53

Example: Verify Certificate Validity | 54

Requirements | 54

Overview | 54

Configuration | 54

Verification | 55

Delete a Loaded CRL (CLI Procedure) | 55

Validate Certificate | 56

Digital Certificate Validation | 56

Example: Validate Digital Certificate by Configuring Policy OIDs on an SRX Series Firewall | 62

Requirements | 62

Overview | 63

Configuration | 63

Verification | 64

Dynamic Update of Trusted CA Certificates | 67

Dynamic Update of Trusted CA Certificates | 67

Configure Dynamic Update of Trusted CA Certificates | 69

Check connectivity to the CDN server | 69

Enable automatic download of default trusted CA certificates | 70

Provide custom configuration for automatic download of default trusted CA certificates | 71

Download default trusted CA certificates explicitly | 72

Check the download status of default trusted CA certificates | 73

Deactivate automatic download of trusted CA certificates | 74

ACME Protocol | 75

What is ACME Protocol | 75

Enroll Local Certificate Using Let's Encrypt Server | 76

Manual Re-Enroll Local Certificate | 78

Delete ACME Account | 78

Configure Multiple Certificate Types to Establish IKE and IPsec SA | 79

Requirements | 79

Overview | 80

Topology | 80

Configuration | 81

Verification | 91

Configuration Statements and Operational Commands | 103

Junos CLI Reference Overview | 103

About This Guide

Use this guide to configure, monitor, and manage the public key infrastructure (PKI) on Juniper Networks devices using Junos OS. Use the PKI for secure data exchange, identity verification, and mutual authentication by using digital certificates.

Configure PKI in Junos OS

1. Create CA Profile. See ["Certificate Authority" on page 16](#).

- Define CA Profile Attributes: Create a CA profile to specify the CA settings, including the CA identity and any additional attributes required.
- Specify Enrollment Parameters: Configure the enrollment retry value and the time interval between attempts to automatically enroll the CA certificates online.
- Set Revocation Check: Specify the Certificate Revocation List (CRL) refresh interval and URL for revocation checks.

2. Generate Certificate. See ["Self-Signed Digital Certificates" on page 24](#).

- Generate Certificate Request: Generate a public or private keypair and then create the certificate request using the keypair.
- Send Certificate Request: Send the certificate request to the CA administrator through email or an out-of-band method. Specify an email address for the CA administrator if needed.

3. Load CA and Local Certificates. See ["Enroll Certificate" on page 30](#).

- Load CA Certificate: Load the CA certificate from an external file and associate it with the configured CA profile.
- Load Local Certificate: Load the local certificate into local storage from the specified external file, ensuring proper linkage with the private or public keypair.

4. Configure IPsec VPN with Certificates. See ["Configure Multiple Certificate Types to Establish IKE and IPsec SA " on page 79](#).

- Define IKE Policy and Gateway: Configure the IKE policy and gateway to use RSA-Signature authentication method and the local and CA certificates.

1

PART

Public Key Infrastructure

[What is PKI | 2](#)

[PKI Certificates | 15](#)

[Manage PKI Certificates | 30](#)

[ACME Protocol | 75](#)

[Configure Multiple Certificate Types to Establish IKE and IPsec SA | 79](#)

[Configuration Statements and Operational Commands | 103](#)

CHAPTER 1

What is PKI

IN THIS CHAPTER

- [PKI Overview | 2](#)
- [PKI Fundamentals | 5](#)
- [PKI in Junos OS | 10](#)
- [PKI Components in Junos OS | 12](#)

PKI Overview

SUMMARY

Learn about PKI, PKI support in Junos OS, and understand the benefits of PKI.

IN THIS SECTION

- [Introduction to PKI | 2](#)
- [How PKI Works | 3](#)
- [Benefits of PKI | 4](#)

Introduction to PKI

Public key infrastructure (PKI) provides a way of verifying the identity of a remote site by using a digital certificate. PKI uses a certificate authority (CA) to validate your information and to sign it with a digital signature such that neither your information nor the signature can be modified. Once signed, the information becomes a digital certificate. Devices that receive a digital certificate can verify the information in the certificate by validating the signature using public key cryptography.

The PKI provides an infrastructure for digital certificate management and consists of:

- Registration Authority (RA) that verifies the identities of entities, authorizes their certificate requests, and generates unique asymmetric key pairs (unless the users' certificate requests already contain public keys)

- Certificate Authority (CA) that issues corresponding digital certificates for the requesting entities.
- A certificate revocation list (CRL) identifying the certificates that are no longer valid. Each entity possessing the authentic public key of a CA can verify the certificates issued by that CA.

How PKI Works

PKI supports the distribution and identification of public encryption keys, enabling users to both securely exchange data over networks such as the Internet and verify the identity of the other party.

Figure 1 on page 3 shows how the authentication happens between two users using the public and private key .

Figure 1: Public Key Infrastructure

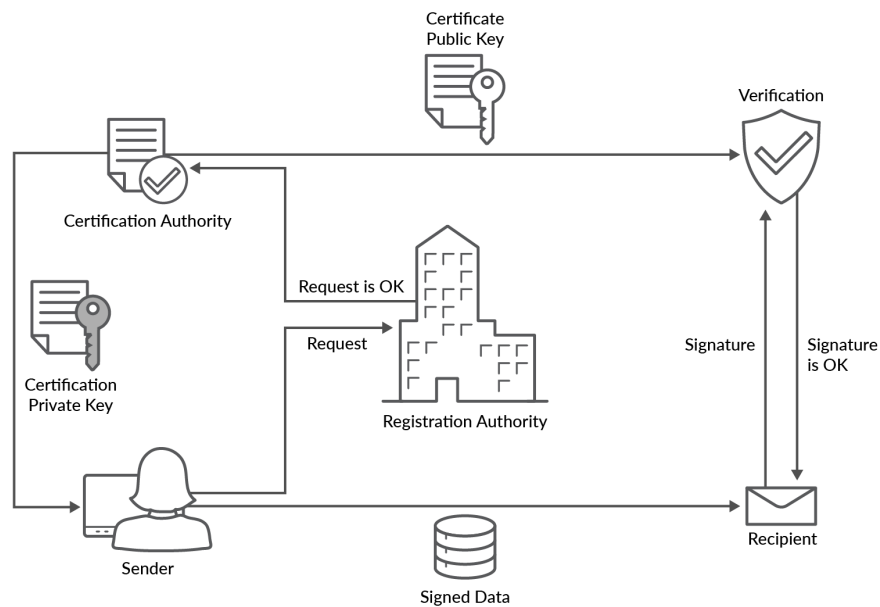


Table 1: Public Key Infrastructure

PKI Key Components	Description
Certificate Authority (CA)	A trusted third-party organization that creates, enrolls, validates, and revokes digital certificates. The CA guarantees a user's identity and issues public and private keys for message encryption and decryption.

Table 1: Public Key Infrastructure (Continued)

PKI Key Components	Description
Registration Authority (RA)	Verifies the identities of entities, authorizes their certificate requests, and generates unique asymmetric key pairs unless the users' certificate requests already contain public keys.
Digital Certificates	These are electronic documents that contain information about the entity to which they are issued, such as a VPN gateway. They are signed by the CA to ensure their authenticity and integrity.
Public and Private Keys	A pair of keys used in public key cryptography. The public key is used for encryption, while the private key is used for decryption. These keys are generated simultaneously and are linked mathematically.
Certificate Life Cycle Management	This includes phases such as generation of public/private keys and identity information, enrollment (request and retrieval), usage within Internet Key Exchange (IKE), certificate validation and revocation checks, and certificate renewal.
IKE and PKI	During IKE Phase 1 setup, a certificate can identify the peer by IP address, fully qualified domain name (FQDN), user fully qualified domain name (U-FQDN), or distinguished name (DN). The IKE ID is added into the <i>SubjectAlternativeName</i> field of the certificate.

Benefits of PKI

- **Enhanced Security:** PKI provides robust security by using asymmetric cryptography, which is more secure than symmetric cryptography. The use of public and private keys ensures that data encrypted with a public key can only be decrypted with the corresponding private key.
- **Trust Hierarchy:** PKI establishes a trust hierarchy through the use of Certificate Authorities (CAs), Registration Authorities (RAs), and Certificate Repositories. This hierarchy ensures that all entities within the network trust each other based on their certificates and the CA that issued them.

- **Data Integrity:** Digital certificates issued by PKI ensure the integrity of data by providing a way to verify the authenticity of the sender and the data itself. This prevents tampering or alteration of data during transmission.
- **Scalability:** PKI is scalable and can be used in large networks with multiple entities. It supports various standards like X.509 and Public Key Cryptography Standards (PKCS), making it versatile and adaptable to different network configurations.
- **Ease of Management:** While setting up a PKI requires some initial configuration, it simplifies the management of digital certificates and keys. This makes it easier to manage and maintain secure connections across the network.

PKI Fundamentals

SUMMARY

Learn about PKI fundamentals such as private/public key and digital certificate. Find out how to verify and monitor the digital certificates.

IN THIS SECTION

- [Digital Certificate | 5](#)
- [Certificate Authority | 6](#)
- [Private/Public Key Pair | 6](#)
- [Certificate Enrollment Options | 6](#)
- [Certificate Revocation Options | 6](#)
- [Certificate Request Types | 7](#)
- [Certificate Signatures and Verification | 7](#)
- [Certificate Validation | 8](#)

This topic describes the overview of public key infrastructure (PKI) and includes the following sections:

Digital Certificate

A digital certificate is an electronic file that verifies the identity of the certificate's holder to protect data exchanged online. 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. Alternatively, you can use a self-signed certificate to attest to your identity.

A key pair is a critical element of a digital certificate implementation. The public key is included in the local digital certificate and the private key is used to decrypt data received from peers.

Certificates have a finite lifetime and are defined by a start time and an end time. The certificate becomes invalid when the life time expires. When the certificate expires, a certificate renewal or a new certificate request is required.

Certificate Authority

A CA is a trusted third-party organization that creates, enrolls, validates, and revokes digital certificates. The CA guarantees a user's identity and issues public and private keys for message encryption and decryption (coding and decoding). A CA also generates certificate revocation lists (CRLs) which are lists of revoked certificates.

Private/Public Key Pair

When setting up a PKI, you must include Public and private keys that are generated in pairs and linked mathematically.

When request for the certificate, you must include the public key in the certificate enrollment request. The public key will be included in the granted certificate and the private key is kept on the requesting device. A message encrypted with the public key can be decrypted by using the corresponding private key. The private-public key pair is also used for creating digital signatures.

Certificate Enrollment Options

You can request a CA digital certificate either online or manually:

- Manual certificate enrollment—This process includes generation of a PKCS10 request, submission to the certificate authority (CA), retrieval of the signed certificate, and manually loading of the certificate into the Junos OS device as the local certificate.
- Online certificate enrollment—You can use either Certificate Management Protocol version 2 (CMPv2) or Simple Certificate Enrollment Protocol (SCEP) for online certificate enrollment.

Certificate Revocation Options

- Certificate revocation list (or CRL)—Certificate authority (CA) periodically publishes a list of revoked certificate using a certificate revocation list (CRL). The CRL contains the list of digital certificates with serial numbers that have been canceled before their expiration date.
- Online Certificate Status Protocol (OCSP)—OCSP is used to check the revocation status of X509 certificates. The OCSP provides revocation status on certificates in real time and is useful in time-sensitive situations such as bank transactions and stock trades

Certificate Request Types

Public Key Infrastructure (PKI) allows users to authenticate each other using digital certificates issued by CA. PKI Uses X.509, Public Key Cryptography Standards (PKCS) to define the standard formats for certificates and their use. In PKI, an applicant uses a certificate signing request (CSR) to apply for a digital certificate to a certificate authority (CA). The request can be in one of the standard:

- Public-Key Cryptography Standard # (PKCS#) (PKCS7, PKCS10, PKCS11, PKCS12)
- x509-signaturere.

Certificate Signatures and Verification

A digital certificate is an electronic means for verifying your identity through a trusted third party, known as a certificate authority (CA). Alternatively, you can use a self-signed certificate to attest to your identity.

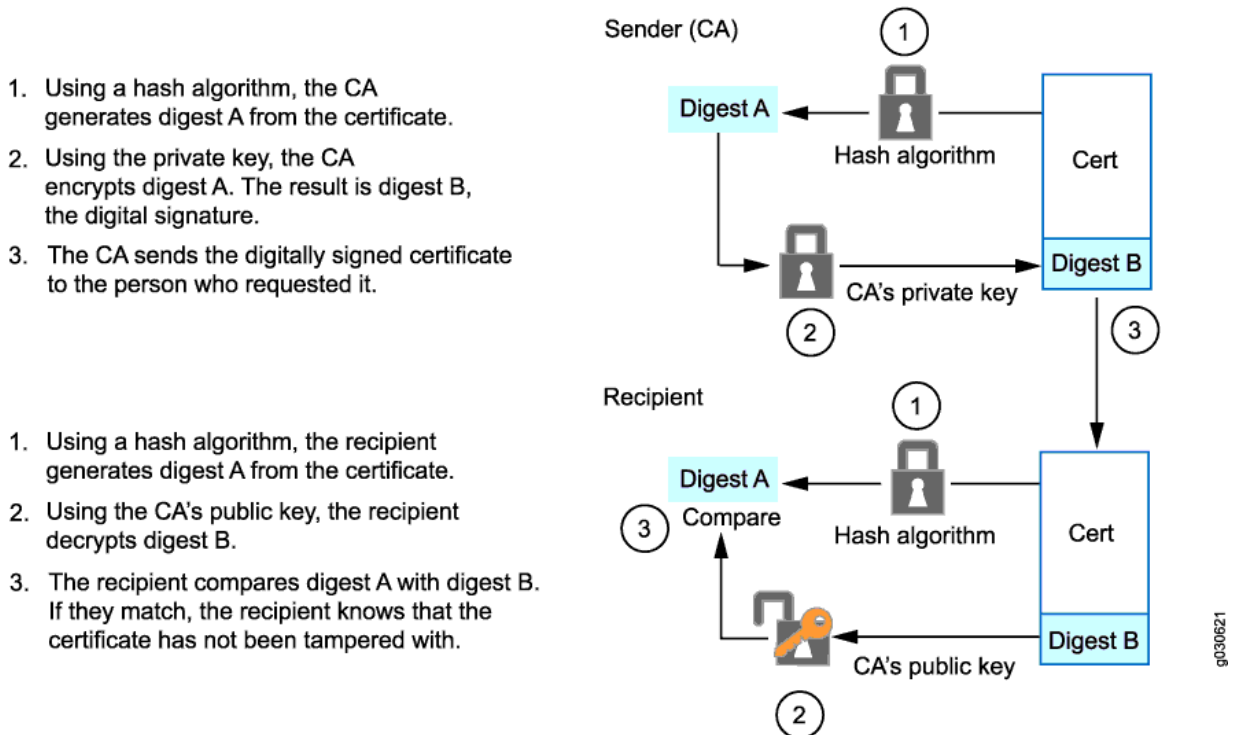
The CA server you use can be owned and operated by an independent CA or by your own organization, in which case you become your own CA. If you use an independent CA, you must contact them for the addresses of their CA and certificate revocation list (CRL) servers (for obtaining certificates and CRLs) and for the information they require when submitting personal certificate requests. When you are your own CA, you determine this information yourself.

The CA that issues a certificate uses a hash algorithm to generate a digest, and then “signs” the certificate by encrypting the digest with its private key. The result is a digital signature. The CA then makes the digitally signed certificate available for download to the person who requested it. [Figure 2 on page 8](#) illustrates this process.

The recipient of the certificate generates another digest by applying the same hash algorithm to the certificate file, then uses the CA's public key to decrypt the digital signature. By comparing the decrypted digest with the digest just generated, the recipient can confirm the integrity of the CA's signature and, by extension, the integrity of the accompanying certificate. [Figure 2 on page 8](#) illustrates this process.

A certificate is considered valid if the digital signature can be verified and the serial number of the certificate is not listed in a certificate revocation list.

Figure 2: Digital Signature Verification



When Digital Signature Algorithm (DSA) signatures are used, the SHA-1 hash algorithm is used to generate the digest. When Rivest-Shamir-Adleman (RSA) signatures are used, SHA-1 is the default hash algorithm used to generate the digest; you can specify the SHA-256 hash algorithm with the digest option of the request security pki generate-certificate-request or request security pki local-certificate generate-self-signed commands. When Elliptic Curve Digital Signature Algorithm (ECDSA) signatures are used, the SHA-256 hash algorithm is used for ECDSA-256 signatures and the SHA-384 hash algorithm is used for ECDSA-384 signatures.

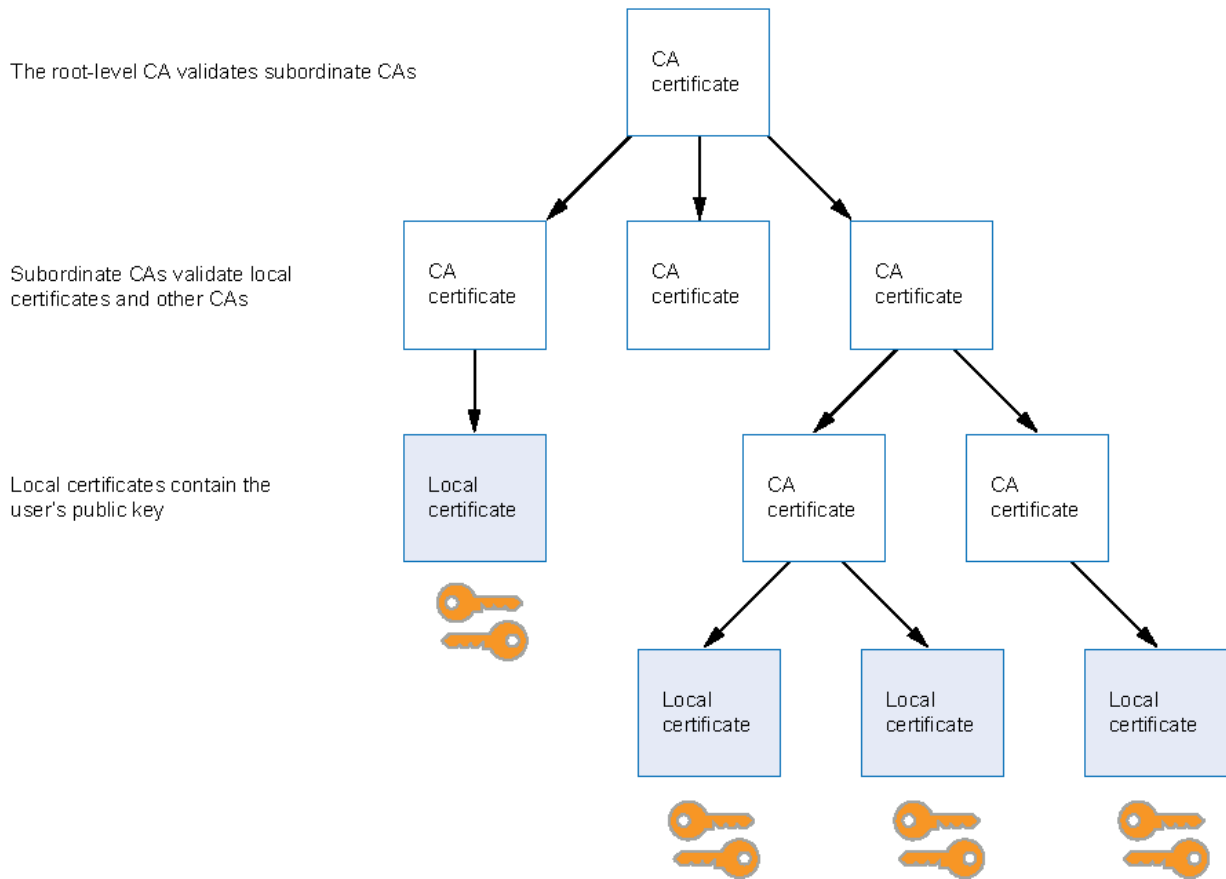
Starting in Junos OS Release 18.1R3, the default hash algorithm that is used for validating automatically and manually generated self-signed PKI certificates is Secure Hash Algorithm 256 (SHA-256). Prior to Junos OS Release 18.1R3, SHA-1 is used as default hash algorithm.

Certificate Validation

To verify the trustworthiness of a certificate, you must be able to track a path of certified certificate authorities (CAs) from the one issuing your local certificate to the root authority of a CA domain. Public key infrastructure (PKI) refers to the hierarchical structure of trust required for the successful implementation of public key cryptography.

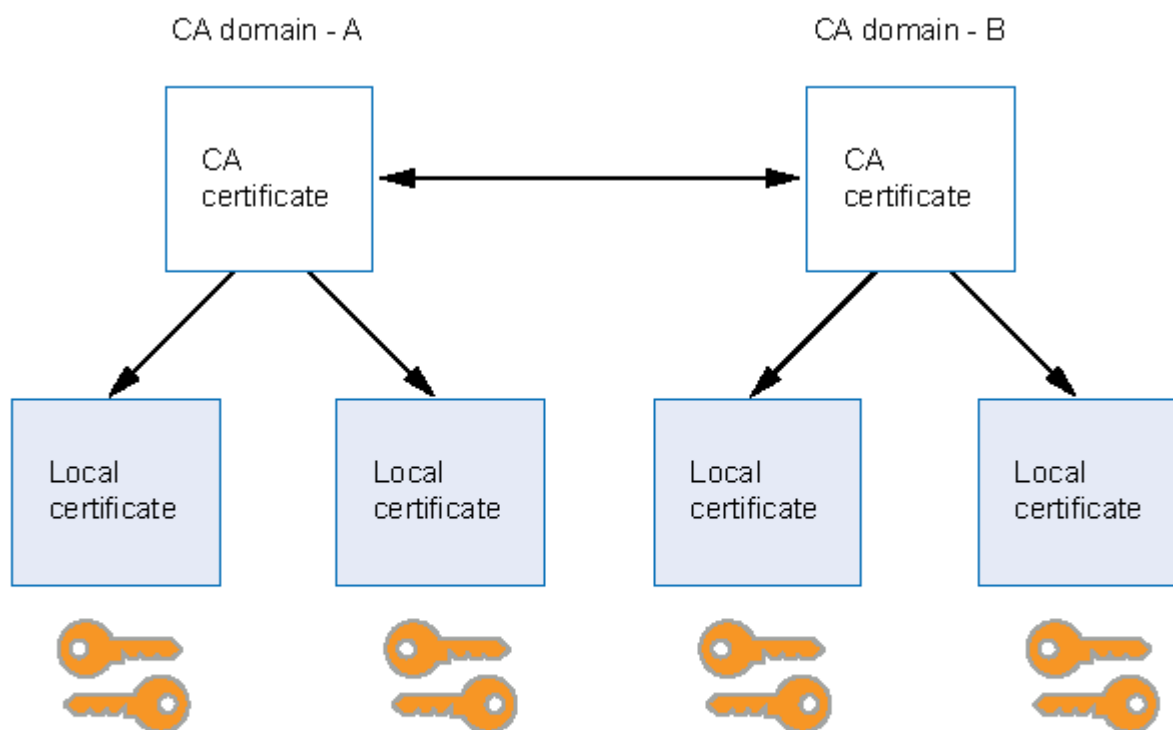
Figure 3 on page 9 shows the structure of a single-domain certificate authority with multiple hierarchy levels.

Figure 3: PKI Hierarchy of Trust—CA Domain



If certificates are used solely within an organization, that organization can have its own CA domain within which a company CA issues and validates certificates for its employees. If that organization later wants its employees to exchange their certificates with certificates from another CA domain (for example, with employees at another organization that has its own CA domain), the two CAs can develop cross-certification by agreeing to trust the authority of each other. In this case, the PKI structure does not extend vertically but does extend horizontally. See [Figure 4 on page 10](#).

Figure 4: Cross-Certification



Users in the CA domain A can use their certificates and key pairs with users in CA domain B because the CA's have cross-certified each other.

PKI in Junos OS

SUMMARY

Learn about the Junos OS applications that require PKI and the basic elements of PKI.

IN THIS SECTION

- [PKI Applications Overview | 10](#)
- [Basic Elements of PKI in Junos OS | 11](#)

PKI Applications Overview

The Junos OS uses public/private keys in the following areas:

- SSH/SCP for secure command-line interface [CLI]-based administration

- Secure Sockets Layer (SSL) for secure Web-based administration and for https-based webauth for user authentication
- Internet Key Exchange (IKE) for IPsec VPN tunnels

Note the following points:

- Currently Junos OS supports only IKE using public key infrastructure (PKI) certificates for public key validation.
- The SSH and SCP are used exclusively for system administration and depends on the use of out-of-band fingerprints for public key identity binding and validation. Details on SSH are not covered in this topic.

Basic Elements of PKI in Junos OS

Junos OS supports three specific types of PKI objects:

- Private/public key pair
- Certificates
 - Local certificate—The local certificate contains the public key and identity information for the Juniper Networks device. The Juniper Networks device owns the associated private key. This certificate is generated based on a certificate request from the Juniper Networks device.
 - Pending certificate — A pending certificate contains a key pair and identity information that is generated into a PKCS10 certificate request and manually sent to a certificate authority (CA). While the Juniper Networks device waits for the certificate from the CA, the existing object (key pair and the certificate request) is tagged as a certificate request or pending certificate.
 - CA certificate — When the certificate is issued by the CA and loaded into the Junos OS device, the pending certificate is replaced by the newly generated local certificate. All other certificates loaded into the device are considered CA certificates.
- Certificate revocation lists (CRLs)

Note the following points about certificates:

- Local certificates are generally used when a Junos OS device has VPNs in more than one administrative domain.
- All PKI objects are stored in a separate partition of persistent memory, apart from the Junos OS image and the system's general configuration.

- Each PKI object has a unique name or certificate-ID given to it when it is created and maintains that ID until its deletion. You can view the certificate-ID by using the `show security pki local-certificate` command.
- A certificate cannot be copied from a device under most circumstances. The private key on a device must be generated on that device only, and it should never be viewed or saved from that device. So PKCS12 files (which contain a certificate with the public key and the associated private key) are not supported on Junos OS devices.
- CA certificates validate the certificates received by the IKE peer. If the certificate is valid, then it is verified in the CRL to see whether the certificate has been revoked.

Each CA certificate includes a CA profile configuration that stores the following information:

- CA identity, which is typically the domain name of the CA
- E-mail address for sending the certificate requests directly to the CA
- Revocation settings:
 - Revocation check enable/disable option
 - Disabling of revocation check in case of CRL download failure.
 - Location of CRL Distribution Point (CDP) (for manual URL setting)
 - CRL refresh interval

PKI Components in Junos OS

SUMMARY

Learn about PKI components and understand how to manage PKI in Junos OS.

IN THIS SECTION

- [PKI Management and Implementation | 13](#)
- [Internet Key Exchange | 13](#)
- [Trusted CA Group | 13](#)
- [Cryptographic Key Handling Overview | 14](#)

This topic includes the following sections:

PKI Management and Implementation

The minimum PKI elements required for certificate-based authentication in Junos OS are:

- CA certificates and authority configuration.
- Local certificates including the device's identity (example: IKE ID type and value) and private and public keys
- Certificate validation through a CRL.

Junos OS supports three different types of PKI objects:

Internet Key Exchange

The procedure for digitally signing messages sent between two participants in an Internet Key Exchange (IKE) session is similar to digital certificate verification, with the following differences:

- Instead of making a digest from the CA certificate, the sender makes it from the data in the IP packet payload.
- Instead of using the CA's public-private key pair, the participants use the sender's public-private key pair.

Trusted CA Group

A Certificate Authority (CA) is a trusted third party responsible for issuing and revoking certificates. You can group multiple CAs (CA profiles) in one trusted CA group for a given topology. These certificates are used to establish connection between two endpoints. To establish IKE or IPsec, both the endpoints must trust the same CA. If either of the endpoints are unable to validate the certificate using their respective trusted CA (ca-profile) or trusted CA group, the connection is not established.

For example, there are two endpoints, endpoint A and endpoint B are trying to establish a secure connection. When endpoint B presents its certificate to endpoint A, the endpoint A will check if the certificate is valid. The CA of the endpoint A verifies the signed certificate that the endpoint B is using to get authorized. When `trusted-ca` or `trusted-ca-group` is configured, the device will only use the CA profiles added in this `trusted-ca-group` or the CA profile configured under `trusted-ca` to validate the certificate coming from endpoint B. If the certificate is verified as valid, the connection is allowed, else the connection is rejected.

Benefits:

- For any incoming connection request, only the certificate issued by that particular trusted CA of that endpoint gets validated. If not, the authorization will reject establishing the connection.

Cryptographic Key Handling Overview

With cryptographic key handling, persistent keys are stored in the memory of the device without any attempt to alter them. While the internal memory device is not directly accessible to a potential adversary, those who require a second layer of defense can enable special handling for cryptographic keys. When enabled, the cryptographic key handling encrypts keys when not immediately in use, performs error detection when copying a key from one memory location to another, and overwrites the memory location of a key with a random bit pattern when the key is no longer in use. Keys are also protected when they are stored in the flash memory of the device. Enabling cryptographic key handling feature does not cause any externally observable change in the behavior of the device, and the device continues to interoperate with the other devices.

A cryptographic administrator can enable and disable the cryptographic self-test functions; however, the security administrator can modify the behavior of the cryptographic self-test functions such as configuring periodic self-tests or selecting a subset of cryptographic self-tests.

The following persistent keys are currently under the management of IKE and PKI:

- IKE preshared keys (IKE PSKs)
- PKI private keys
- Manual VPN keys

CHAPTER 2

PKI Certificates

IN THIS CHAPTER

- [Digital Certificates | 15](#)
- [Certificate Authority | 16](#)
- [Self-Signed Digital Certificates | 24](#)

Digital Certificates

SUMMARY

Learn about the digital certificates. Find out how to configure digital certificates.

IN THIS SECTION

- [Generate Digital Certificates Manually: Configuration Overview | 16](#)

A digital certificate is an electronic means for verifying your identity through a trusted third party, known as a certificate authority (CA). Alternatively, you can use a self-signed certificate to attest to your identity.

Manual certificate processing includes generation of a PKCS10 request, submission to the CA, retrieval of the signed certificate, and manually loading the certificate into the Juniper Networks device. Based on your deployment environment, you can use either SCEP or CMPv2 for online certificate enrollment.

To use a digital certificate to authenticate your identity when establishing a secure VPN connection, you must first do the following:

- Obtain a CA certificate from which you intend to obtain a local certificate, and then load the CA certificate onto the device. The CA certificate can contain a CRL to identify invalid certificates.
- Obtain a local certificate from the CA whose CA certificate you have previously loaded, and then load the local certificate in the device. The local certificate establishes the identity of the Juniper Networks device with each tunnel connection.

Generate Digital Certificates Manually: Configuration Overview

To obtain digital certificates manually:

1. Generate a key pair on the device. See ["Self-Signed Digital Certificates" on page 24.](#)
2. Create a CA profile or profiles containing information specific to a CA. See ["Example: Configure a CA Profile" on page 21.](#)
3. Generate the CSR for the local certificate and send it to the CA server. See ["Example: Manually Generate a CSR for the Local Certificate and Send it to the CA Server" on page 40 .](#)
4. Load the certificate onto the device. See ["Example: Load CA and Local Certificates Manually" on page 42.](#)
5. Configure automatic reenrollment. See ["Example: Using SCEP to Automatically Renew a Local Certificate" on page 35.](#)
6. If necessary, load the certificate's CRL on the device. See ["Example: Manually Load a CRL onto the Device" on page 48.](#)
7. If necessary, configure the CA profile with CRL locations. See ["Example: Configure a Certificate Authority Profile with CRL Locations" on page 52](#)

Certificate Authority

SUMMARY

Learn about the certificate authority (CA) and understand how to manage CA.

IN THIS SECTION

- [Configure a Trusted CA Group | 17](#)
- [Certificate Authority Profiles | 20](#)
- [Example: Configure a CA Profile | 21](#)
- [Example: Configure an IPv6 address as the Source Address for a CA Profile | 23](#)

A certificate authority (CA) profile define every parameter associated with a specific certificate to establish secure connection between two endpoints. The profiles specify which certificates to use, how to verify certificate revocation status, and how that status constrains access.

Configure a Trusted CA Group

IN THIS SECTION

- [Create a Trusted CA Group for a List of CA Profiles | 17](#)
- [Delete a CA Profile from a Trusted CA Group | 18](#)
- [Delete a Trusted CA Group | 19](#)

This section describes the procedure to create a trusted CA group for a list of CA profiles and delete a trusted CA group.

Create a Trusted CA Group for a List of CA Profiles

You can configure and assign a trusted CA group to authorize an entity. When a peer tries to establish a connection with a client, only the certificate issued by that particular trusted CA of that entity gets validated. The device validates if the issuer of the certificate and the one presenting the certificate belongs to the same client network. If the issuer and the presenter belong to the same client network then the connection is established. If not, the connection will not be established.

Before you begin, you must have a list of all the CA profiles you want to add to the trusted group.

In this example, we are creating three CA profiles named `orgA-ca-profile`, `orgB-ca-profile`, and `orgC-ca-profile` and associating the following CA identifiers `ca-profile1`, `ca-profile2`, and `ca-profile3` for the respective profiles. You can group all the three CA profiles to belong to a trusted CA group `orgABC-trusted-ca-group`.

You can configure a maximum of 20 CA profiles for a trusted CA group.

1. Create CA profiles and associate CA identifiers to the profile.

```
[edit]
user@host# set security pki ca-profile orgA-ca-profile ca-identity ca-profile1
user@host# set security pki ca-profile orgB-ca-profile ca-identity ca-profile2
user@host# set security pki ca-profile orgC-ca-profile ca-identity ca-profile3
```


2. Group the CA profiles under a trusted CA group.

```
[edit]
set security pki trusted-ca-group orgABC-trusted-ca-group ca-profiles [orgA-ca-profile orgB-
ca-profile orgC-ca-profile]
```

3. Commit the configuration when you are done configuring the CA profiles and the trusted CA groups.

```
[edit]
user@host# commit
```

To view the CA profiles and the trusted CA groups configured on your device, run `show security pki` command.

```
user@host# show security pki
ca-profile orgA-ca-profile {
    ca-identity ca-profile1;
}
ca-profile orgB-ca-profile {
    ca-identity ca-profile2;
}
ca-profile orgC-ca-profile {
    ca-identity ca-profile3;
}
trusted-ca-group orgABC-trusted-ca-group {
    ca-profiles [ orgA-ca-profile orgB-ca-profile orgC-ca-profile ];
}
```

The `show security pki` command displays all the CA profiles that are grouped under the `orgABC-trusted-ca-group`.

Delete a CA Profile from a Trusted CA Group

You can delete a specific CA profile in a trusted CA group or you can delete the trusted CA group itself.

For example, if you want to delete a CA profile named `orgC-ca-profile` from a trusted CA group `orgABC-trusted-ca-group`, configured on your device as shown in ["Configure a Trusted CA Group" on page 17](#) topic perform the following steps:

1. Delete a CA profile from the trusted CA group.

```
[edit]
user@host# delete security pki trusted-ca-group orgABC-trusted-ca-group ca-profiles orgC-ca-profile
```

2. If you are done deleting the CA profile from the trusted CA group, commit the configuration.

```
[edit]
user@host# commit
```

To view the orgC-ca-profile being deleted from the orgABC-trusted-ca-group , run the `show security pki` command.

```
user@host# show security pki
ca-profile orgA-ca-profile {
    ca-identity ca-profile1;
}
ca-profile orgB-ca-profile {
    ca-identity ca-profile2;
}
trusted-ca-group orgABC-trusted-ca-group {
    ca-profiles [ orgA-ca-profile orgB-ca-profile ];
}
```

The output does not display the orgC-ca-profile profile as it is deleted from the trusted CA group.

Delete a Trusted CA Group

An entity can support many trusted CA groups and you can delete any trusted CA group for an entity.

For example, if you want to delete a trusted CA group named orgABC-trusted-ca-group, configured on your device as shown in ["Configure a Trusted CA Group" on page 17](#) topic perform the following steps:

1. Delete a trusted CA group.

```
[edit]
user@host# delete security pki trusted-ca-group orgABC-trusted-ca-group
```

2. If you are done deleting the CA profile from the trusted CA group, commit the configuration.

```
[edit]
user@host# commit
```

To view the orgABC-trusted-ca-group being deleted from the entity , run the show security pki command.

```
user@host# show security pki
ca-profile orgA-ca-profile {
    ca-identity ca-profile1;
}
ca-profile orgB-ca-profile {
    ca-identity ca-profile2;
}
```

The output does not display the orgABC-trusted-ca-group as it is deleted from the entity.

Certificate Authority Profiles

A certificate authority (CA) profile configuration contains information specific to a CA. You can have multiple CA profiles on an SRX Series Firewall. For example, you might have one profile for orgA and one for orgB. Each profile is associated with a CA certificate. If you want to load a new CA certificate without removing the older one then create a new CA profile (for example, Microsoft-2008).

Starting with Junos OS Release 18.1R1, the CA server can be an IPv6 CA server.

The PKI module supports IPv6 address format to enable the use of SRX Series Firewalls in networks where IPv6 is the only protocol used.

A CA issues digital certificates, which helps to establish secure connection between two endpoints through certificate validation. You can group multiple CA profiles in one trusted CA group for a given topology. These certificates are used to establish a connection between two endpoints. To establish IKE or IPsec, both the endpoints must trust the same CA. If either of the endpoints are unable to validate the certificate using their respective trusted CA (ca-profile) or trusted CA group, the connection is not established. A minimum of one CA profile is mandatory to create a trusted CA group and maximum of 20 CAs are allowed in one trusted CA group. Any CA from a particular group can validate the certificate for that particular endpoint.

Starting with Junos OS Release 18.1R1, validation of a configured IKE peer can be done with a specified CA server or group of CA servers. A group of trusted CA servers can be created with the trusted-ca-group configuration statement at the [edit security pki] hierarchy level; one or multiple CA profiles can be specified. The trusted CA server is bound to the IKE policy configuration for the peer at [edit security ike policy *policy* certificate] hierarchy level.

If proxy profile is configured in CA profile, the device connects to the proxy host instead of the CA server while certificate enrollment, verification or revocation. The proxy host communicates with the CA server with the requests from the device, and then relay the response to the device.

CA proxy profile supports SCEP, CMPv2, and OCSP protocols.

CA proxy profile is supported only on HTTP and is not supported on HTTPS protocol.

Example: Configure a CA Profile

IN THIS SECTION

- [Requirements | 21](#)
- [Overview | 21](#)
- [Configuration | 22](#)
- [Verification | 23](#)

This example shows how to configure a CA profile.

Requirements

No special configuration beyond device initialization is required before configuring this feature.

Overview

In this example, you create a CA profile called `ca-profile-ipsec` with CA identity `microsoft-2008`. You then create proxy profile to the CA profile. The configuration specifies that the CRL be refreshed every 48 hours, and the location to retrieve the CRL is `http://www.my-ca.com`. Within the example, you set the enrollment retry value to 20. (The default retry value is 10.)

Automatic certificate polling is set to every 30 minutes. If you configure retry only without configuring a retry interval, then the default retry interval is 900 seconds (or 15 minutes). If you do not configure retry or a retry interval, then there is no polling.

Configuration

IN THIS SECTION

- [Procedure | 22](#)

Procedure

Step-by-Step Procedure

To configure a CA profile:

1. Create a CA profile.

```
[edit]
user@host# set security pki ca-profile ca-profile-ipsec ca-identity microsoft-2008
user@host#
```

2. Optionally, configure the proxy profile to the CA profile.

```
[edit]
user@host# set security pki ca-profile ca-profile-ipsec proxy-profile px-profile
```

Public key infrastructure (PKI) uses proxy profile configured at the system-level. The proxy profile being used in the CA profile must be configured at the `[edit services proxy]` hierarchy. There can be more than one proxy profile configured under `[edit services proxy]` hierarchy. Each CA profile is referred to the most one such proxy profile. You can configure host and port of the proxy profile at the `[edit system services proxy]` hierarchy.

3. Create a revocation check to specify a method for checking certificate revocation.

```
[edit]
user@host# set security pki ca-profile ca-profile-ipsec ca-identity microsoft-2008 revocation-
check crl
```

4. Set the refresh interval, in hours, to specify the frequency in which to update the CRL. The default values are next-update time in CRL, or 1 week, if no next-update time is specified.

```
[edit]
user@host# set security pki ca-profile ca-profile-ipsec ca-identity microsoft-2008 revocation-
check crl refresh-interval 48 url http://www.my-ca.com/my-crl.crl
```

5. Specify the enrollment retry value.

```
[edit]
user@host# set security pki ca-profile ca-profile-ipsec enrollment retry 20
```

6. Specify the time interval in seconds between attempts to automatically enroll the CA certificate online.

```
[edit]
user@host# set security pki ca-profile ca-profile-ipsec enrollment retry-interval 1800
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security pki` command.

Example: Configure an IPv6 address as the Source Address for a CA Profile

This example shows how to configure an IPv6 address as the source address for a CA profile.

No special configuration beyond device initialization is required before configuring this feature.

In this example, create a CA profile called `orgA-ca-profile` with CA identity `v6-ca` and set the source address of the CA profile to be an IPv6 address, such as `2001:db8:0:f101::1`. You can configure the enrollment URL to accept an IPv6 address `http://[2002:db8:0:f101::1]:/.../`.

1. Create a CA profile.

```
[edit]
user@host# set security pki ca-profile orgA-ca-profile ca-identity v6_ca
```

2. Configure the source address of the CA profile to be an IPv6 address.

```
[edit]
user@host# set security pki ca-profile v6_ca source-address 2001:db8:0:f101::1
```

3. Specify the enrollment parameters for the CA.

```
[edit]
user@host# set security pki ca-profile v6_ca enrollment url http://[2002:db8:0:f101::1]:/.../
```

4. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Self-Signed Digital Certificates

SUMMARY

Learn about the self-signed digital certificate and find out how to manage the self-signed digital certificate.

IN THIS SECTION

- [Self-Signed Certificates | 25](#)
- [Example: Generate a Public-Private Key Pair | 26](#)
- [Example: Manually Generate Self-Signed Certificates | 27](#)
- [Using Automatically Generated Self-Signed Certificates \(CLI Procedure\) | 28](#)

A self-signed certificate is a certificate that is signed by the same entity who created it rather than by a Certificate Authority (CA). Junos OS provides two methods for generating a self-signed certificate - automatic generation and manual generation.

Self-Signed Certificates

A self-signed certificate is a certificate that is signed by its creator rather than by a Certificate Authority (CA).

Self-signed certificates allow for use of SSL-based (Secure Sockets Layer) services without requiring that the user or administrator to undertake the considerable task of obtaining an identity certificate signed by a CA.

Self-signed certificates do not provide additional security as do those generated by CAs. This is because a client cannot verify that the server connected to is the one advertised in the certificate.

Junos OS provides two methods for generating a self-signed certificate:

- Automatic generation

In this case, the creator of the certificate is the Juniper Networks device. An automatically generated self-signed certificate is configured on the device by default.

After the device is initialized, it checks for the presence of an automatically generated self-signed certificate. If it does not find one, the device generates one and saves it in the file system.

- Manual generation

In this case, you create the self-signed certificate for the device.

At any time, you can use the CLI to generate a self-signed certificate. These certificates are also used to gain access to SSL services.

Self-signed certificates are valid for five years from the time they were generated.

An automatically generated self-signed certificate allows for use of SSL-based services without requiring that the administrator obtain an identity certificate signed by a CA.

A self-signed certificate that is automatically generated by the device is similar to a Secure Shell (SSH) host key. It is stored in the file system, not as part of the configuration. It persists when the device is rebooted, and it is preserved when a `request system snapshot` command is issued.

A self-signed certificate that you manually generate allows for use of SSL-based services without requiring that you obtain an identity certificate signed by a CA. A manually generated self-signed certificate is one example of a public key infrastructure (PKI) local certificate. As is true of all PKI local certificates, manually generated self-signed certificates are stored in the file system.

Example: Generate a Public-Private Key Pair

IN THIS SECTION

- [Requirements | 26](#)
- [Overview | 26](#)
- [Configuration | 26](#)
- [Verification | 27](#)

This example shows how to generate a public-private key pair.

Requirements

No special configuration beyond device initialization is required before configuring this feature.

Overview

In this example, you generate a public-private key pair named self-cert.

Configuration

IN THIS SECTION

- [Procedure | 26](#)

Procedure

Step-by-Step Procedure

To generate a public-private key pair:

- Create a certificate key pair.

```
user@host> request security pki generate-key-pair certificate-id self-cert
```

Verification

After the public-private key pair is generated, the Juniper Networks device displays the following:

```
generated key pair ca-ipsec, key size 1024 bits
```

Example: Manually Generate Self-Signed Certificates

IN THIS SECTION

- [Requirements | 27](#)
- [Overview | 27](#)
- [Configuration | 27](#)
- [Verification | 28](#)

This example shows how to generate self-signed certificates manually.

Requirements

Before you begin, generate a public private key pair. See ["Digital Certificates" on page 15](#).

Overview

For a manually generated self-signed certificate, you specify the distinguished name (DN) when you create it. For an automatically generated self-signed certificate, the system supplies the DN, identifying itself as the creator.

In this example, you generate a self-signed certificate with the e-mail address as `mholmes@example.net`. You specify a certificate-id of `self-cert` to be referenced by web management.

Configuration

IN THIS SECTION

- [Procedure | 28](#)

Procedure

Step-by-Step Procedure

To generate the self-signed certificate manually, enter the following command in operational mode:

```
user@host> request security pki local-certificate generate-self-signed certificate-id self-cert  
subject CN=abc domain-name example.net ip-address 1.2.3.4 email mholmes@example.net
```

To specify the manually generated self-signed certificate for Web management HTTPS services, enter the following command in configuration mode:

```
[edit]  
user@host# set system services web-management https local-certificate self-cert
```

Verification

To verify the certificate is properly generated and loaded, enter the following command in operational mode:

```
user@host> show security pki local-certificate
```

Notice the Certificate identifier information for Issued to, validity, algorithm, and keypair location details in the displayed output.

To verify the certificate that is associated with the web management, enter the following command in configuration mode:

```
user@host# show system services web-management https local-certificate
```

Using Automatically Generated Self-Signed Certificates (CLI Procedure)

After the device is initialized, it checks for the presence of a self-signed certificate. If a self-signed certificate is not present, the device automatically generates one. If the device is rebooted, a self-signed certificate is automatically generated at boot time.

To check the system-generated certificate, run the following command in operational mode:

```
user@host> show security pki local-certificate system-generated
```

Notice the Certificate identifier details in the output. It displays the following details distinguished name (DN) for the automatically generated certificate:

- CN = *device serial number*
- CN = system generated
- CN = self-signed

Use the following command in configuration mode to specify the automatically generated self-signed certificate to be used for Web management HTTPS services:

```
[edit]  
user@host# set system services web-management https system-generated-certificate
```

Use the following operational command to delete the automatically generated self-signed certificate:

```
user@host# exit  
user@host> clear security pki local-certificate system-generated
```

After you delete the system-generated self-signed certificate, the device automatically generates a new one and saves it in the file system.

CHAPTER 3

Manage PKI Certificates

IN THIS CHAPTER

- [Enroll Certificate | 30](#)
- [Certificate Revocation | 44](#)
- [Validate Certificate | 56](#)
- [Dynamic Update of Trusted CA Certificates | 67](#)

Enroll Certificate

SUMMARY

Learn about how to enroll and manage various PKI certificates.

IN THIS SECTION

- [Enroll Digital Certificates Online: Configuration Overview | 31](#)
- [Online CA Certificate Enrollment | 31](#)
- [Local Certificate Requests | 31](#)
- [Enroll a CA Certificate Online Using SCEP | 32](#)
- [Example: Enroll a Local Certificate Online Using SCEP | 33](#)
- [Example: Using SCEP to Automatically Renew a Local Certificate | 35](#)
- [CMPv2 and SCEP Certificate Enrollment | 37](#)
- [Certificate Enrollment with CMPv2 | 38](#)
- [Example: Manually Generate a CSR for the Local Certificate and Send it to the CA Server | 40](#)

- [Example: Load CA and Local Certificates Manually](#) | 42
- [Delete Certificates \(CLI Procedure\)](#) | 44

A certificate authority (CA) issues digital certificates, which helps to establish a secure connection between two endpoints through certificate validation. The following topics describe how to configure CA certificates online or local using Simple Certificate Enrollment Protocol (SCEP):

Enroll Digital Certificates Online: Configuration Overview

You can use either Certificate Management Protocol version 2 (CMPv2) or Simple Certificate Enrollment Protocol (SCEP) to enroll digital certificates. To enroll a certificate online:

1. Generate a key pair on the device. See ["Self-Signed Digital Certificates" on page 24](#).
2. Create a CA profile or profiles containing information specific to a CA. See ["Example: Configure a CA Profile" on page 21](#).
3. For SCEP only, enroll the CA certificate. See ["Enroll a CA Certificate Online Using SCEP" on page 32](#).
4. Enroll the local certificate from the CA whose CA certificate you have previously loaded. See ["Example: Enroll a Local Certificate Online Using SCEP" on page 33](#).
5. Configure automatic reenrollment. See ["Example: Using SCEP to Automatically Renew a Local Certificate" on page 35](#).

Online CA Certificate Enrollment

With Simple Certificate Enrollment Protocol (SCEP), you can configure your Juniper Networks device to obtain a certificate authority (CA) certificate online and start the online enrollment for the specified certificate ID. The CA public key verifies certificates from remote peers.

Local Certificate Requests

When you create a local certificate request, the device generates an end entity certificate in PKCS #10 format from a key pair you previously generated using the same certificate ID.

A subject name is associated with the local certificate request in the form of a common name (CN), organizational unit (OU), organization (O), locality (L), state (ST), country (C), and domain component (DC). Additionally, a subject alternative name is associated in the following form:

- IP address

- E-mail address
- Fully qualified domain name (FQDN)

Specify the subject name in the distinguished name format in quotation marks, including the domain component (DC), common name (CN), serial number (SN), organizational unit name (OU), organization name (O), locality (L), state (ST), and country (C).

Some CAs do not support an e-mail address as the domain name in a certificate. If you do not include an e-mail address in the local certificate request, you cannot use an e-mail address as the local IKE ID when configuring the device as a dynamic peer. Instead, you can use a fully qualified domain name (if it is in the local certificate), or you can leave the local ID field empty. If you do not specify a local ID for a dynamic peer, enter the *hostname.domain-name* of that peer on the device at the other end of the IPsec tunnel in the peer ID field.

Enroll a CA Certificate Online Using SCEP

Before you begin:

1. Generate a public and private key pair.
2. Create a CA profile.

To enroll a CA certificate online:

1. Retrieve the CA certificate online using SCEP. (The attributes required to reach the CA server are obtained from the defined CA profile.)

```
user@host> request security pki ca-certificate enroll ca-profile ca-profile-ipsec
```

The command is processed synchronously to provide the fingerprint of the received CA certificate.

```
Fingerprint:
e6:fa:d6:da:e8:8d:d3:00:e8:59:12:e1:2c:b9:3c:c0:9d:6c:8f:8d (sha1)
82:e2:dc:ea:48:4c:08:9a:fd:b5:24:b0:db:c3:ba:59 (md5)
Do you want to load the above CA certificate ? [yes,no]
```

2. Confirm that the correct certificate is loaded. The CA certificate is loaded only when you type **yes** at the CLI prompt.

For more information on the certificate, such as the bit length of the key pair, use the command `show security pki ca-certificate`.

Example: Enroll a Local Certificate Online Using SCEP

IN THIS SECTION

- [Requirements | 33](#)
- [Overview | 33](#)
- [Configuration | 34](#)
- [Verification | 35](#)

This example shows how to enroll a local certificate online using Simple Certificate Enrollment Protocol (SCEP).

Requirements

Before you begin:

- Generate a public and private key pair. See ["Self-Signed Digital Certificates" on page 24](#).
- Configure a certificate authority profile. See ["Example: Configure a CA Profile" on page 21](#).
- For SCEP, enroll the CA certificate. See ["Enroll a CA Certificate Online Using SCEP" on page 32](#).

Overview

In this example, you configure your Juniper Networks device to obtain a local certificate online and start the online enrollment for the specified certificate ID with SCEP. You specify the URL path to the CA server in the CA profile name `ca-profile-ipsec`.

You use the `request security pki local-certificate enroll scep` command to start the online enrollment for the specified certificate ID. (Starting in Junos OS Release 15.1X49-D40 and Junos OS Release 17.3R1, the `scep` keyword is supported and required.) You must specify the CA profile name (for example, `ca-profile-ipsec`), the certificate ID corresponding to a previously generated key-pair (for example, `qqq`), and the following information:

- The challenge password provided by the CA administrator for certificate enrollment and reenrollment.
- At least one of the following values:
 - The domain name to identify the certificate owner in IKE negotiations—for example, `qqq.example.net`.

- The identity of the certificate owner for IKE negotiation with the e-mail statement—for example, `qqq@example.net`.
- The IP address if the device is configured for a static IP address—for example, `10.10.10.10`.

Specify the subject name in the distinguished name format in quotation marks, including the domain component (DC), common name (CN), serial number (SN), organizational unit name (OU), organization name (O), locality (L), state (ST), and country (C).

Once the device certificate is obtained and the online enrollment begins for the certificate ID. The command is processed asynchronously.

Configuration

IN THIS SECTION

- [Procedure | 34](#)

Procedure

Step-by-Step Procedure

To enroll a local certificate online:

1. Specify the CA profile.

```
[edit]
user@host# set security pki ca-profile ca-profile-ipsec enrollment url path-to-ca-server
```

2. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

3. Initiate the enrollment process by running the operational mode command.

```
user@host> request security pki local-certificate enroll scep ca-profile ca-profile-ipsec
certificate-id qq challenge-password ca-provided-password domain-name qq.example.net email
```

```
qqq@example.net ip-address 10.10.10.10 subject DC=example, CN=router3, SN, OU=marketing,
O=example, L=sunnyvale, ST=california, C=us
```

If you define SN in the subject field without the serial number, then the serial number is read directly from the device and added to the certificate signing request (CSR).

Starting in Junos OS Release 19.4R2, a warning message ECDSA Keypair not supported with SCEP for cert_id *<certificate id>* is displayed when you try to enroll local certificate using an Elliptic Curve Digital Signature Algorithm (ECDSA) key with Simple Certificate Enrollment Protocol (SCEP) as ECDSA key is not supported with SCEP.

Verification

To verify the configuration is working properly, enter the `show security pki` command.

Example: Using SCEP to Automatically Renew a Local Certificate

IN THIS SECTION

- [Requirements | 35](#)
- [Overview | 36](#)
- [Configuration | 36](#)
- [Verification | 37](#)

You can use either Certificate Management Protocol version 2 (CMPv2) or Simple Certificate Enrollment Protocol (SCEP) to enroll digital certificates. This example shows how to renew the local certificates automatically using SCEP.

Requirements

Before you begin:

- Obtain a certificate either on line or manually. See ["Digital Certificates" on page 15](#).
- Obtain a local certificate. See ["Example: Enroll a Local Certificate Online Using SCEP" on page 33](#).

Overview

You can enable the device to automatically renew certificates that were acquired by online enrollment or loaded manually. Automatic certificate renewal saves you from having to remember to renew certificates on the device before they expire, and helps to maintain valid certificates at all times.

Automatic certificate renewal is disabled by default. You can enable automatic certificate renewal and configure the device to automatically send out a request to reenroll a certificate before it expires. You can specify when the certificate reenrollment request is to be sent; the trigger for reenrollment is the percentage of the certificate's lifetime that remains before expiration. For example, if the renewal request is to be sent when the certificate's remaining lifetime is 10 percent, then configure 10 for the reenrollment trigger.

For this feature to work, the device must be able to reach the CA server, and the certificate must be present on the device during the renewal process. Furthermore, you must also ensure that the CA issuing the certificate can return the same DN. The CA must not modify the subject name or alternate subject name extension in the new certificate.

You can enable and disable automatic SCEP certificate renewal either for all SCEP certificates or on a per-certificate basis. You use the `set security pki auto-re-enrollment scep` command to enable and configure certificate reenrollment. In this example, you specify the certificate ID of the CA certificate as `ca-ipsec` and set the CA profile name associated with the certificate to `ca-profile-ipsec`. You set the challenge password for the CA certificate to the challenge password provided by the CA administrator; this password must be the same one configured previously for the CA. You also set the percentage for the reenrollment trigger to 10. During automatic reenrollment, the Juniper Networks device by default uses the existing key pair. A good security practice is to regenerate a new key pair for reenrollment. To generate a new key pair, use the `re-generate-keypair` command.

Configuration

IN THIS SECTION

- [Procedure | 36](#)

Procedure

Step-by-Step Procedure

To enable and configure local certificate reenrollment:

1. To enable and configure certificate reenrollment.

```
[edit]
user@host# set security pki auto-re-enrollment scep certificate-id ca-ipsec ca-profile-name
ca-profile-ipsec challenge-password ca-provided-password re-enroll-trigger-time-percentage
10 re-generate-keypair
```

Starting in Junos OS Release 15.1X49-D40 and Junos OS Release 17.3R1, the scep keyword is supported and required.

2. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security pki local-certificate detail` operational mode command.

CMPv2 and SCEP Certificate Enrollment

Based on your deployment environment, you can use either Certificate Management Protocol version 2 (CMPv2) or Simple Certificate Enrollment Protocol (SCEP) for online certificate enrollment. This topic describes some of the basic differences between the two protocols.

[Table 2 on page 37](#) describes the differences between the CMPv2 and SCEP certificate enrollment protocols.

Table 2: Comparison of CMPv2 and SCEP Certificate Enrollment

Attribute	CMPv2	SCEP
Supported certificate types:	DSA, ECDSA, and RSA	RSA only
Supported standards	RFCs 4210 and 4211	Internet Engineering Task Force draft

Certificate enrollment and reenrollment requests and responses differ between CMPv2 and SCEP. With CMPv2, there is no separate command to enroll CA certificates. With SCEP, you enroll CA certificates with the `request security pki ca-certificate enroll` command and specify the CA profile. A CA profile must be configured with either CMPv2 or SCEP.

Certificate Enrollment with CMPv2

IN THIS SECTION

- [Certificate Enrollment and Reenrollment Messages | 38](#)
- [End-Entity Certificate with Issuer CA Certificate | 38](#)
- [End-Entity Certificate with CA Certificate Chain | 39](#)

The request `security pki local-certificate enroll cmpv2` command uses CMPv2 to enroll a local digital certificate online. This command loads both end-entity and CA certificates based on the CA server configuration. The CA profile must be created prior to CA certificate enrollment because the enrollment URL is extracted from the CA profile.

This topic describes certificate enrollment with the CMPv2 protocol.

Certificate Enrollment and Reenrollment Messages

The CMPv2 protocol mainly involves certificate enrollment and reenrollment operations. The certificate enrollment process includes Initialization Request and Initialization Response messages, while certificate reenrollment includes Key Update Request and Key Update Response messages.

CMPv2 server responds back with Initialization Response (IP). The response contains end-entity certificate along with optional CA certificates. The message integrity and message authenticity of Initialization Response can be verified using shared-secret-information according to RFC 4210. The Initialization Response can also be verified using issuer CA public-key. Before you re-enroll an end-entity certificate, you must have a valid CA certificate enrolled on the device.

The Initialization Response or Key Update Response message can contain an issuer CA certificate or a chain of CA certificates. The CA certificates received in the responses are treated as trusted CA certificates and stored in the receiving device if they are not already present in the trusted CA store. These CA certificates are later used for end-entity certificate validation.

We do not support CA certificate re-enrollment. If a CA certificate expires, you must unenroll the current CA certificate and enroll it back again.

End-Entity Certificate with Issuer CA Certificate

In a simple scenario, the Initialization Response message might contain only an end-entity certificate, in which case the CA information is provided separately. The certificate is stored in the end-entity certificate store.

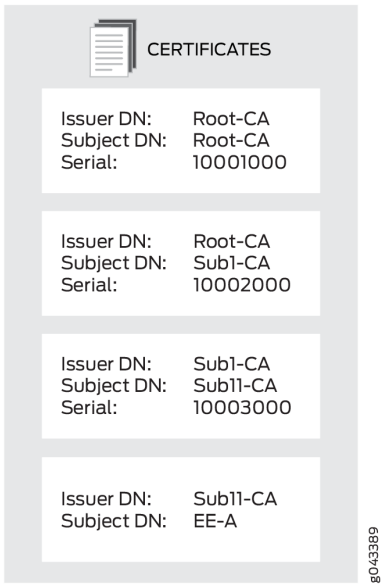
The Initialization Response message can contain an end-entity certificate as well as a self-signed issuer CA certificate. The end-entity certificate is first stored in the certificate store, and then the CA certificate is checked. If the CA certificate is found and the subject distinguished name (DN) of the CA certificate in the Initialization Response message matches the issuer DN of the end-entity certificate, the CA certificate is stored in the CA certificate store for the CA profile name specified in the CMPv2 certificate enrollment command. If the CA certificate already exists in the CA certificate store, no action is taken.

End-Entity Certificate with CA Certificate Chain

In many deployments, the end-entity certificate is issued by an intermediate CA in a certificate chain. In this case, the Initialization Response message can contain the end-entity certificate along with a list of CA certificates in the chain. The intermediate CA certificates and the self-signed root CA certificates are all required to validate the end-entity certificate. The CA chain might also be needed to validate certificates received from peer devices with similar hierarchies. The following section describes how certificates in the CA chain are stored.

In [Figure 5 on page 39](#), the Initialization Response message includes the end-entity certificate and three CA certificates in a certificate chain.

Figure 5: End-Entity Certificate with CA Certificate Chain



The end-entity certificate is stored in the end-entity certificate store. Each CA certificate needs a CA profile. The CA certificate with the subject DN Sub11-CA is the first CA in the chain and is the issuer of the end-entity certificate. It is stored in the CA profile that is specified with the CMPv2 certificate enrollment command.

Each of the remaining CA certificates in the chain is checked for its presence in the CA store. If a CA certificate is not present in the CA store, it is saved and a CA profile is created for it. The new CA profile name is created using the least significant 16 digits of the CA certificate serial number. If the serial number is longer than 16 digits, the most significant digits beyond 16 digits are truncated. If the serial number is shorter than 16 digits, the remaining most significant digits are filled with 0s. For example, if the serial number is 11111000100010001000, then the CA profile name is 1000100010001000. If the serial number is 10001000, then the CA profile name is 0000000010001000.

It is possible that multiple certificate serial numbers can have the same least significant 16 digits. In that case, -00 is appended to the profile name to create a unique CA profile name; additional CA profile names are created by incrementing the appended number, from -01 up to -99. For example, CA profile names can be 1000100010001000, 1000100010001000-00, and 1000100010001000-01.

Example: Manually Generate a CSR for the Local Certificate and Send it to the CA Server

IN THIS SECTION

- [Requirements | 40](#)
- [Overview | 40](#)
- [Configuration | 41](#)
- [Verification | 41](#)

This example shows how to generate a certificate signing request manually.

Requirements

Generate a public and private key. See ["Self-Signed Digital Certificates" on page 24](#).

Overview

In this example, you generate a certificate request using the certificate ID of a public-private key pair you previously generated (ca-ipsec). Then you specify the domain name (example.net) and the associated common name (abc). The certificate request is displayed in PEM format.

You copy the generated certificate request and paste it into the appropriate field at the CA website to obtain a local certificate. (Refer to the CA server documentation to determine where to paste the certificate request.) When the PKCS #10 content is displayed, the MD5 hash and SHA-1 hash of the PKCS #10 file is also displayed.

Configuration

IN THIS SECTION

- [Procedure | 41](#)

Procedure

Step-by-Step Procedure

To generate a local certificate manually:

- Specify certificate ID, domain name, and common name.

```
user@host> request security pki generate-certificate-request certificate-id ca-ipsec domain-
name example.net subject CN=abc
```

Verification

To view the certificate signing request, enter the `show security pki certificate-request detail` command.

```
Certificate identifier: ca-ipsec
Certificate version: 1
Issued to: CN = abc
Public key algorithm: rsaEncryption(1024 bits)
30:81:89:02:81:81:00:da:ea:cd:3a:49:1f:b7:33:3c:c5:50:fb:57
de:17:34:1c:51:9b:7b:1c:e9:1c:74:86:69:a4:36:77:13:a7:10:0e
52:f4:2b:52:39:07:15:3f:39:f5:49:d6:86:70:4b:a6:2d:73:b6:68
39:d3:6b:f3:11:67:ee:b4:40:5b:f4:de:a9:a4:0e:11:14:3f:96:84
03:3c:73:c7:75:f5:c4:c2:3f:5b:94:e6:24:aa:e8:2c:54:e6:b5:42
c7:72:1b:25:ca:f3:b9:fa:7f:41:82:6e:76:8b:e6:d7:d2:93:9b:38
fe:fd:71:01:2c:9b:5e:98:3f:0c:ed:a9:2b:a7:fb:02:03:01:00:01
Fingerprint:
0f:e6:2e:fc:6d:52:5d:47:6e:10:1c:ad:a0:8a:4c:b7:cc:97:c6:01 (sha1)
f8:e6:88:53:52:c2:09:43:b7:43:9c:7a:a2:70:98:56 (md5)
```


Example: Load CA and Local Certificates Manually

IN THIS SECTION

- [Requirements | 42](#)
- [Overview | 42](#)
- [Configuration | 43](#)
- [Verification | 43](#)

This example shows how to load CA and local certificates manually.

Requirements

Before you begin:

- Generate a public-private key pair. See ["Self-Signed Digital Certificates" on page 24](#).
- Create a CA profile. See ["Certificate Authority" on page 16](#)
- CA Profile is only required for the CA certificate and not for the local certificate.
- Generate a certificate request. See ["Example: Manually Generate a CSR for the Local Certificate and Send it to the CA Server" on page 40](#).

Overview

In this example, you download the local.cert and ca.cert certificates and save them to the /var/tmp/ directory on the device.

After you download certificates from a CA, you transfer them to the device (for example, using FTP), and then load them.

You can load the following certificate files onto a device running Junos OS:

- A local or end-entity (EE) certificate that identifies your local device. This certificate is your public key.
- A CA certificate that contains the CA's public key.
- A CRL that lists any certificates revoked by the CA.

You can load multiple EE certificates onto the device.

Configuration

IN THIS SECTION

- [Procedure | 43](#)

Procedure

Step-by-Step Procedure

To load the certificate files onto a device:

1. Load the local certificate.

```
[edit]
user@host> request security pki local-certificate load certificate-id local.cert
filename /var/tmp/local.cert
```

2. Load the CA certificate.

```
[edit]
user@host> request security pki ca-certificate load ca-profile ca-profile-ipsec
filename /var/tmp/ca.cert
```

3. Examine the fingerprint of the CA certificate, if it is correct for this CA certificate select yes to accept.

Verification

To verify the certificates loaded properly, enter the `show security pki local-certificate` and `show security pki ca-certificate` commands in operational mode.

```
Fingerprint:
e8:bf:81:6a:cd:26:ad:41:b3:84:55:d9:10:c4:a3:cc:c5:70:f0:7f (sha1)
19:b0:f8:36:e1:80:2c:30:a7:31:79:69:99:b7:56:9c (md5)
Do you want to load this CA certificate ? [yes,no] (no) yes
```

Delete Certificates (CLI Procedure)

You can delete a local or trusted CA certificate that is automatically or manually generated.

Use the following command to delete a local certificate:

```
user@host> clear security pki local certificate certificate-id (certificate-id | all | system-generated )
```

Specify a certificate ID to delete a local certificate with a specific ID, use `all` to delete all local certificates, or specify `system-generated` to delete the automatically generated self-signed certificate.

When you delete an automatically generated self-signed certificate, the device generates a new one.

To delete a CA certificate:

```
user@host> clear security pki ca-certificate ca-profile (ca-profile-name | all)
```

Specify a CA profile to delete a specific CA certificate, or use `all` to delete all CA certificates present in the persistent store.

You are asked for confirmation before a CA certificate can be deleted.

Certificate Revocation

SUMMARY

Learn how to revoke various PKI certificates.

IN THIS SECTION

- [Online Certificate Status Protocol and Certificate Revocation Lists | 45](#)
- [Example: Manually Load a CRL onto the Device | 48](#)
- [Dynamic CRL Download and Verify | 49](#)
- [Example: Configure a Certificate Authority Profile with CRL Locations | 52](#)
- [Example: Verify Certificate Validity | 54](#)

Digital certificates have an expiration date, however, prior to expiration, a certificate may no longer be valid due to many reasons. You can manage certificate revocations and validations locally and by referencing a Certificate Authority (CA) certificate revocation list (CRL).

Online Certificate Status Protocol and Certificate Revocation Lists

IN THIS SECTION

- [Comparison of Online Certificate Status Protocol and Certificate Revocation List | 47](#)

OCSP is used to check the revocation status of X509 certificates. OCSP provides revocation status on certificates in real time and is useful in time-sensitive situations such as bank transactions and stock trades.

The revocation status of a certificate is checked by sending a request to an OCSP server that resides outside of an SRX Series Firewall. Based on the response from the server, the VPN connection is allowed or denied. OCSP responses are not cached on SRX Series Firewalls.

The OCSP server can be the *certificate authority (CA)* that issues a certificate or a designated authorized responder. The location of the OCSP server can be configured manually or extracted from the certificate that is being verified. Requests are sent first to OCSP server locations that are manually configured in CA profiles with the `ocsp url` statement at the [edit security pki ca-profile *profile-name* revocation-check] hierarchy level; up to two locations can be configured for each CA profile. If the first configured OCSP server is not reachable, the request is sent to the second OCSP server. If the second OCSP server is not reachable, the request is then sent to the location in the certificate's AuthorityInfoAccess extension field. The `use-ocsp` option must also be configured, as *certificate revocation list (CRL)* is the default checking method.

SRX Series Firewalls accept only signed OCSP responses from the CA or authorized responder. The response received is validated using trusted certificates. The response is validated as follows:

1. The CA certificate enrolled for the configured CA profile is used to validate the response.
2. The OCSP response might contain a certificate to validate the OCSP response. The received certificate must be signed by a CA certificate enrolled in the SRX Series Firewall. After the received certificate is validated by the CA certificate, it is used to validate the OCSP response.

The response from the OCSP server can be signed by different CAs. The following scenarios are supported:

- The CA server that issues the end entity certificate for a device also signs the OCSP revocation status response. The SRX Series Firewall verifies the OCSP response signature using the CA certificate enrolled in the SRX Series Firewall. After the OCSP response is validated, the certificate revocation status is checked.
- An authorized responder signs the OCSP revocation status response. The certificate for the authorized responder and the end entity certificate being verified must be issued by the same CA. The authorized responder is first verified using the CA certificate enrolled in the SRX Series Firewall. The OCSP response is validated using the responder's CA certificate. The SRX Series Firewall then uses the OCSP response to check the revocation status of the end entity certificate.
- There are different CA signers for the end entity certificate being verified and the OCSP response. The OCSP response is signed by a CA in the certificate chain for the end entity certificate being verified. (All peers participating in an IKE negotiation need to have at least one common trusted CA in their respective certificate chains.) The OCSP responder's CA is verified using a CA in the certificate chain. After validating the responder CA certificate, the OCSP response is validated using the responder's CA certificate.

To prevent replay attacks, a *nonce* payload can be sent in an OCSP request. Nonce payloads are sent by default unless it is explicitly disabled. If enabled, the SRX Series Firewall expects the OCSP response to contain a nonce payload, otherwise the revocation check fails. If OCSP responders are not capable of responding with a nonce payload, then the nonce payload must be disabled on the SRX Series Firewall.

In the normal course of business, certificates are revoked for various reasons. You might wish to revoke a certificate if you suspect that it has been compromised, for example, or when a certificate holder leaves the company.

You can manage certificate revocations and validations in two ways:

- Locally— This is a limited solution.
- By referencing a Certificate Authority (CA) certificate revocation list (CRL)— You can automatically access the CRL online at intervals you specify or at the default interval set by the CA.

In Phase 1 negotiations, SRX Series Firewall verifies the EE certificate received from the peer during an IKE exchange and uses the CRL to make sure the EE certificate is not revoked by its CA.

If a CRL is not loaded on the device and the peer certificate issuer is a trusted CA:

1. Junos OS retrieves the CRL through the configured LDAP or HTTP CRL locations (that is, the CRL Distribution Points (CDP)), if they are defined in the CA profile.

2. If the CRL Distribution Points is not configured in the CA profile, the device uses the CDP extension in a certificate issued by the CA (if present). The certificate issued by the CA can be a certificate enrolled by the administrator or received during the Phase 1 negotiation.

If the EE certificate is not issued by a root CA, the certificates of each intermediate CAs goes through the same verification and revocation check. The CRL of the root CA is used to check if the certificate issued by the root CA is revoked. If the CDP is not configured in the root CA profile, the device uses the CDP extension in the certificate issued by the CA (if present).

The CRL distribution point extension (.cdp) in an X509 certificate can be added to either an HTTP URL or an LDAP URL.

If the certificate does not contain a certificate distribution point extension, and you cannot automatically retrieve the CRL through Lightweight Directory Access Protocol (LDAP) or Hypertext Transfer Protocol (HTTP), you can retrieve a CRL manually and load that in the device.

Local certificates are being validated against certificate revocation list (CRL) even when CRL check is disabled. This can be stopped by disabling the CRL check through the Public Key Infrastructure (PKI) configuration. When CRL check is disabled, PKI will not validate local certificate against CRL.

Comparison of Online Certificate Status Protocol and Certificate Revocation List

Online Certificate Status Protocol (OCSP) and certificate revocation list (CRL) can both be used to check the revocation status of a certificate. There are advantages and disadvantages to each method.

- OCSP provides certificate status in real time, while CRL uses cached data. For time-sensitive applications, OCSP is the preferred approach.
- CRL checking is faster because lookup for certificate status is done on information cached on the VPN device. OCSP requires time to obtain the revocation status from an external server.
- CRL requires additional memory to store the revocation list received from a CRL server. OCSP does not require additional memory to save the revocation status of certificates.
- OCSP requires that the OCSP server be available at all times. CRL can use cached data to check the revocation status of certificates when the server is unreachable.

On MX Series and SRX Series Firewalls, CRL is the default method used to check the revocation status of a certificate.

SEE ALSO

| [Certificate Validation](#)

Example: Manually Load a CRL onto the Device

IN THIS SECTION

- [Requirements | 48](#)
- [Overview | 48](#)
- [Configuration | 49](#)
- [Verification | 49](#)

This example shows how to load a CRL manually onto the device.

Requirements

Before you begin:

1. Generate a public and private key pair. See ["Self-Signed Digital Certificates" on page 24](#).
2. Generate a certificate request. See ["Example: Configure a CA Profile" on page 21](#).
3. Configure a certificate authority (CA) profile. See ["Example: Configure a CA Profile" on page 21](#).
4. Load your certificate onto the device. See ["Example: Load CA and Local Certificates Manually" on page 42](#).

Overview

You can load a CRL manually, or you can have the device load it automatically, when you verify certificate validity. To load a CRL manually, you obtain the CRL from a CA and transfer it to the device (for example, using FTP).

In this example, you load a CRL certificate called `revoke.crl` from the `/var/tmp` directory on the device. The CA profile is called `ca-profile-ipsec`. (Maximum file size is 5 MB.)

If a CRL is already loaded into the `ca-profile` the command `clear security pki crl ca-profile ca-profile-ipsec` must be run first to clear the old CRL.

Configuration

IN THIS SECTION

- Procedure | 49

Procedure

Step-by-Step Procedure

To load a CRL certificate manually:

1. Load a CRL certificate.

```
[edit]
user@host> request security pki crl load ca-profile ca-profile-ipsec filename /var/tmp/
revoke.crl
```

Junos OS supports loading of CA certificates in X509, PKCS #7, DER, or PEM formats.

Verification

To verify the configuration is working properly, enter the `show security pki crl operational mode` command.

Dynamic CRL Download and Verify

Digital certificates are issued for a set period of time and are invalid after the specified expiration date. A CA can revoke an issued certificate by listing it in a certificate revocation list (CRL). During peer certificate validation, the revocation status of a peer certificate is checked by downloading the CRL from a CA server to the local device.

To facilitate the CRL check for the certificates when a CA profile is not configured, dynamic CA profile is created. A dynamic CA profile is automatically created on the local device with the format `dynamic-nnn`.

A dynamic CA profile:

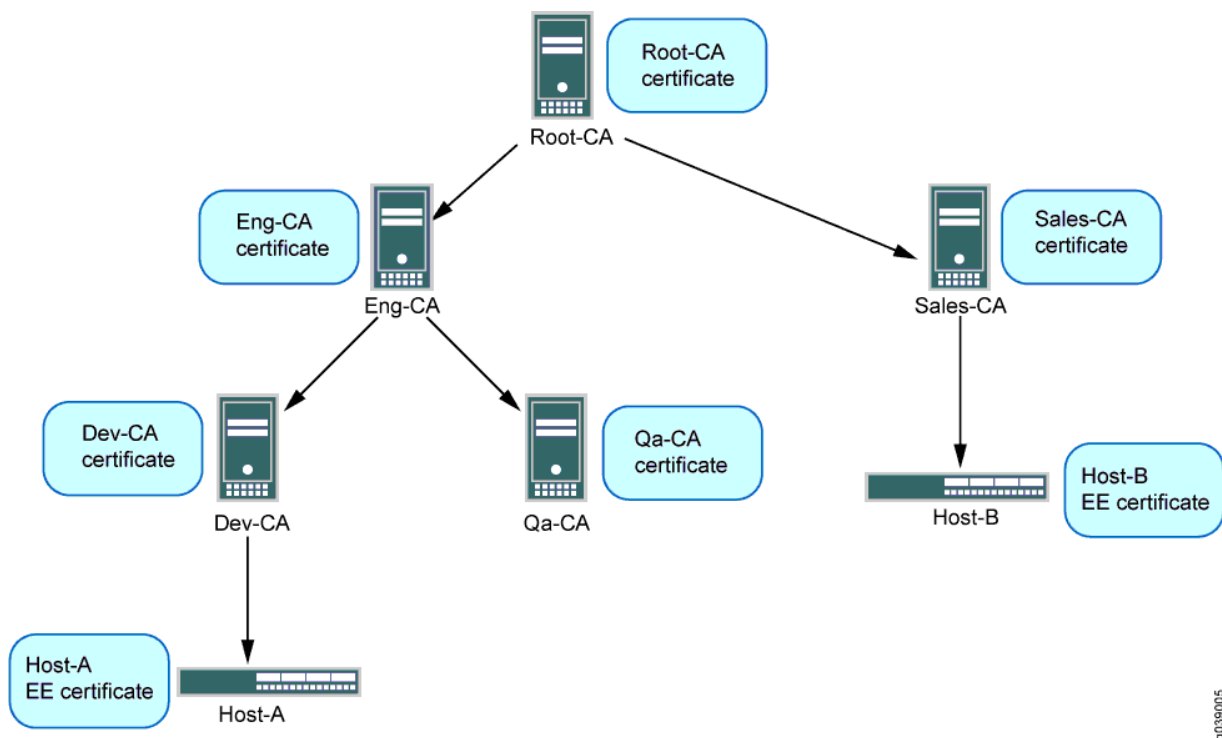
- Allows the local device to download the Dynamic CA and Dynamic CRL (for corresponding CA) as per peer's localcert issuer
- Checks the revocation status of the peer's certificate

A VPN device checks a peer's EE certificate for its revocation status. A VPN device uses the certificate received from its peer to do the following:

- Extract the URL to dynamically download the CA's CRL
- Check the revocation status of the peer's EE certificate

In [Figure 6 on page 50](#), Host-A can use the Sales-CA and EE certificates received from Host-B to dynamically download the CRL for Sales-CA and check the revocation status of Host-B's certificate.

Figure 6: Multilevel Hierarchy for Certificate-Based Authentication



In case of single hierarchy CA servers or CA certificate chain, the local EE certificate and the received peer EE certificate are issued from the same CA server.

Following are some of the SRX Series Firewall behavior based on different configurations:

- If you have configured a SRX Series Firewall with a trusted-ca or trusted-ca-group, then the device does not validate or trust any other CAs.
- If you have defined a CA profile that has a chain of CAs where the SRX Series Firewall only trusts the root CA and peer has a certificate signed by a sub-CA to this root, then Dynamic CA and CRL will be added to the device.

[Table 3 on page 51](#) provides few sample scenarios where Dynamic CA or CRL is not created:

Table 3: Sample Scenarios

Scenario	Condition
Sample scenario 1	In the CA profile, you have defined a trusted CA for <code>ca-profile-name</code> , and you receive a connection from a device that has a certificate signed by a different CA that was not defined as a trusted CA in your CA profile.
Sample scenario 2	You have defined a CA profile that has a chain of CAs where the SRX Series Firewall only trust a sub-CA, and peer has a certificate signed by a level above this sub-CA.

To enable dynamic CA profiles, you must configure the `revocation-check crl` option on a Root-CA profile at the `[edit security pki ca-profile profile-name]` hierarchy level.

The revocation check properties of a Root-CA profile are inherited for dynamic CA profiles. In [Figure 6 on page 50](#), the CA profile configuration on Host-A for Root-CA enables dynamic CA profiles as shown in the following output:

```
admin@host-A# show security
pki {
  ca-profile Root-CA {
    ca-identity Root-CA;
    enrollment {
      url "www.example.net/scep/Root/";
    }
    revocation-check {
      crl;
    }
  }
}
```

A dynamic CA profile is created on Host-A for Sales-CA. Revocation checking is inherited for the Sales-CA dynamic CA profile from Root-CA.

If the `revocation-check disable` statement is configured in a Root-CA profile, dynamic CA profiles are not created and dynamic CRL download and checking is not performed.

The data for CRLs downloaded from dynamic CA profiles are displayed with the `show security pki crl` command in the same way as CRLs downloaded by configured CA profiles. The CRL from a dynamic CA profile is updated periodically as are those for CA profiles that are configured in the device. The peer CA certificate is also required for signature validation of CRL downloaded from CA server.

The CA certificate is required to validate the CRL received from a CA server; therefore, the CA certificate received from a peer is stored on the local device. The received CA certificate from peer is used to validate the CRL and the certificate it issued. Because the received CA certificate is not enrolled by an administrator, the result of a successful certificate verification is not conclusive until the whole certificate chain up to the root CA is verified. The certificate of the root CA must be enrolled by an administrator.

Example: Configure a Certificate Authority Profile with CRL Locations

IN THIS SECTION

- [Requirements | 52](#)
- [Overview | 53](#)
- [Configuration | 53](#)
- [Verification | 53](#)

This example shows how to configure a certificate authority profile with CRL locations.

Requirements

Before you begin:

1. Generate a key pair in the device. See ["Digital Certificates" on page 15](#).
2. Create a CA profile or profiles containing information specific to a CA. See ["Example: Configure a CA Profile" on page 21](#).
3. Obtain a personal certificate from the CA. See ["Example: Manually Generate a CSR for the Local Certificate and Send it to the CA Server" on page 40](#).
4. Load the certificate onto the device. See ["Example: Load CA and Local Certificates Manually" on page 42](#).
5. Configure automatic reenrollment. See [Example: Configuring SecurID User Authentication](#).

6. If necessary, load the certificate's CRL on the device. See ["Example: Manually Load a CRL onto the Device" on page 48](#).

Overview

In this example, you direct the device to check the validity of the CA profile called `my_profile` and, if a CRL did not accompany a CA certificate and is not loaded on the device, to retrieve the CRL from the URL `http://abc/abc-crl.crl`.

Configuration

IN THIS SECTION

- [Procedure | 53](#)

Procedure

Step-by-Step Procedure

To configure certificate using CRL:

1. Specify the CA profile and URL.

```
[edit]
user@host# set security pki ca-profile my_profile revocation-check crl url http://abc/abc-crl.crl
```

2. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security pki operational mode` command.

Example: Verify Certificate Validity

IN THIS SECTION

- [Requirements | 54](#)
- [Overview | 54](#)
- [Configuration | 54](#)
- [Verification | 55](#)

This example shows how to verify the validity of a certificate.

Requirements

No special configuration beyond device initialization is required before configuring this feature.

Overview

In this example, you verify certificates manually to find out whether a certificate has been revoked or whether the CA certificate used to create a local certificate is no longer present on the device.

When you verify certificates manually, the device uses the CA certificate (`ca-cert`) to verify the local certificate (`local.cert`). If the local certificate is valid, and if `revocation-check` is enabled in the CA profile, the device verifies that the CRL is loaded and valid. If the CRL is not loaded and valid, the device downloads the new CRL.

For CA-issued certificates or CA certificates, a DNS must be configured in the device's configuration. The DNS must be able to resolve the host in the distribution CRL and in the CA cert/revocation list url in the ca-profile configuration. Additionally, you must have network reachability to the same host in order for the checks to receive.

Configuration

IN THIS SECTION

- [Procedure | 55](#)

Procedure

Step-by-Step Procedure

To manually verify the validity of a certificate:

1. Verify the validity of a local certificate.

```
[edit]  
user@host> request security pki local-certificate verify certificate-id local.cert
```

2. Verify the validity of a CA certificate.

```
[edit]  
user@host> request security pki ca-certificate verify ca-profile ca-profile-ipsec
```

The associated private key and the signature are also verified.

Verification

To verify the configuration is working properly, enter the `show security pki ca-profile` command.

If an error is returned instead of a positive verification the failure is logged in `pkid`.

Delete a Loaded CRL (CLI Procedure)

You can choose to delete a loaded CRL if you no longer need to use it to manage certificate revocations and validation.

Use the following command to delete a loaded certificate revocation list:

```
user@host> clear security pki crl ca-profile (ca-profile all)
```

Specify a CA profile to delete a CRL associated with the CA identified by the profile, or use `all` to delete all CRLs.

Validate Certificate

SUMMARY

Learn how you can dynamically update the CA certificates.

IN THIS SECTION

- [Digital Certificate Validation | 56](#)
- [Example: Validate Digital Certificate by Configuring Policy OIDs on an SRX Series Firewall | 62](#)

Digital Certificate Validation

IN THIS SECTION

- [Policy Validation | 56](#)
- [Path Length Validation | 59](#)
- [Key Usage | 60](#)
- [Issuer and Subject Distinguished Name Validation | 61](#)

During IKE negotiation, the PKI daemon on an SRX Series Firewall validates X509 certificates received from VPN peers. The certificate validation performed is specified in RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Basic certificate and certificate chain validations include signature and date validation as well as revocation checks. This topic describes additional digital certificate validations performed by the PKI daemon.

Policy Validation

X509 certificates can include optional policy validation fields. If a policy validation field is present, policy validation is performed for the entire certificate chain including the end entity (EE) certificate and intermediate certificate authority (CA) certificates. Policy validation is not applicable to the root certificate. Policy validation ensures that the EE and intermediate CA certificates have a common policy. If no common policy exists for the certificate chain being validated, certificate validation fails.

Prior to policy validation, a certificate chain containing the self-signed root certificate, intermediate CA certificates, and EE certificate must be built. The policy validation starts with the intermediate CA certificate issued by the self-signed root certificate and continues through the EE certificate.

The following optional certificate fields are used for policy validation:

- **policy-oids**
- **requireExplicitPolicy**
- **skipCerts**

These fields are described in the following sections.

Policy OIDs Configured on SRX Series Firewalls

In some situations, it might be desirable to only accept certificates with known policy object identifiers (OIDs) from peers. This optional configuration allows certificate validation to succeed only if the certificate chain received from the peer contains at least one policy OID that is configured on the SRX Series Firewall.

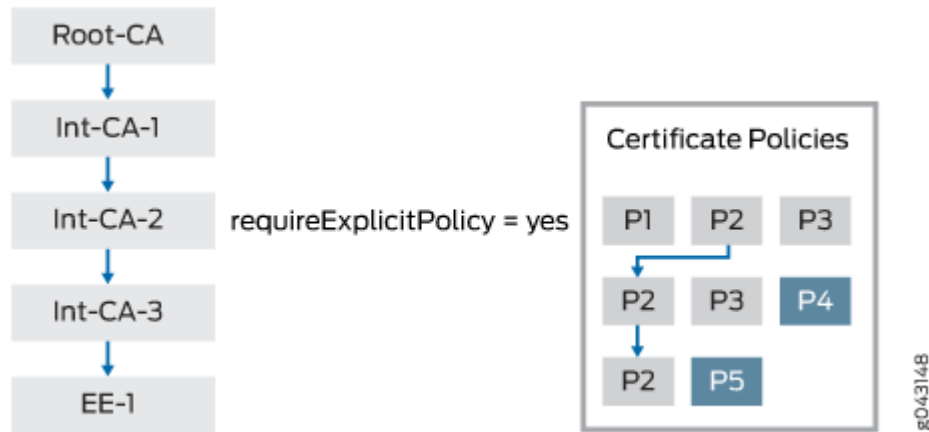
On the SRX Series Firewall, policy OIDs are configured in an IKE policy with the `policy-oids` configuration statement at the `[edit security ike policy policy-name certificate]` hierarchy level. You can configure up to five policy OIDs. For a peer's certificate to be validated successfully, the peer's certificate chain must contain at least one of the policy OIDs configured on the SRX Series Firewall. Note that the **policy-oids** field in a certificate is optional. If you configure policy OIDs on the SRX Series Firewall but the peer's certificate chain does not contain any policy OIDs, certificate validation fails.

No Policy OIDs Configured on SRX Series Firewalls

If no policy OID is configured on the SRX Series Firewall, policy validation starts whenever the **requireExplicitPolicy** field is encountered in the certificate chain. A certificate can contain one or more certificate policy OIDs. For policy validation to succeed, there must be a common policy OID in the certificate chain.

[Figure 7 on page 58](#) shows a certificate chain that consists of certificates for a root CA, three intermediate CAs, and an EE. The CA certificate for Int-CA-2 contains the **requireExplicitPolicy** field; therefore, policy validation starts with Int-CA-2 and continues through EE-1. The certificate for Int-CA-2 contains policy OIDs P1, P2, and P3. The certificate for Int-CA-3 contains policy OIDs P2, P3, and P4. The certificate for EE-1 contains policy OIDs P2 and P5. Because the policy OID P2 is common to the certificates being validated, policy validation succeeds.

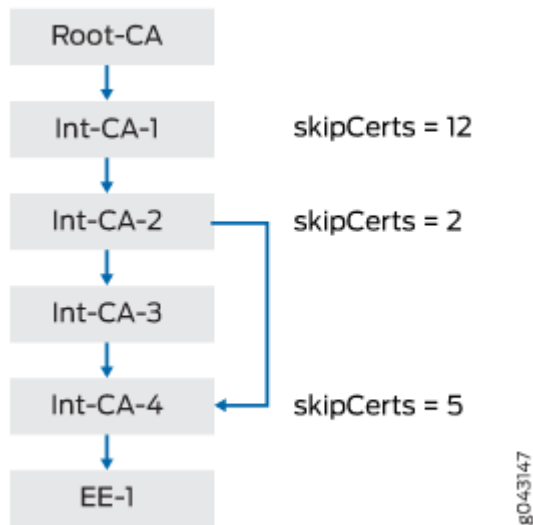
Figure 7: Policy Validation with requireExplicitPolicy Field



The optional **skipCerts** field in an intermediate CA certificate indicates the number of certificates, including the current CA certificate, that are to be excluded from policy validation. If **skipCerts** is 0, policy validation starts from the current certificate. If **skipCerts** is 1, the current certificate is excluded from policy validation. The value of the **skipCerts** field is checked in every intermediate CA certificate. If a **skipCerts** value is encountered that is lower than the current number of certificates being excluded, the lower **skipCerts** value is used.

Figure 8 on page 59 shows a certificate chain consisting of a root CA, four intermediate CAs, and an EE. The **skipCerts** value in Int-CA-1 is 12, which skips 12 certificates including the certificate for Int-CA-1. However, the **skipCerts** value is checked in every intermediate CA certificate in the chain. The **skipCerts** value in Int-CA-2 is 2, which is lower than 12, so now 2 certificates are skipped. The **skipCerts** value in Int-CA-4 is 5, which is greater than 2, so the Int-CA-4 **skipCerts** value is ignored.

Figure 8: Policy Validation with skipCerts Field



When policy OIDs are configured on the SRX Series Firewall, the certificate fields **requireExplicitPolicy** and **skipCerts** are ignored.

Path Length Validation

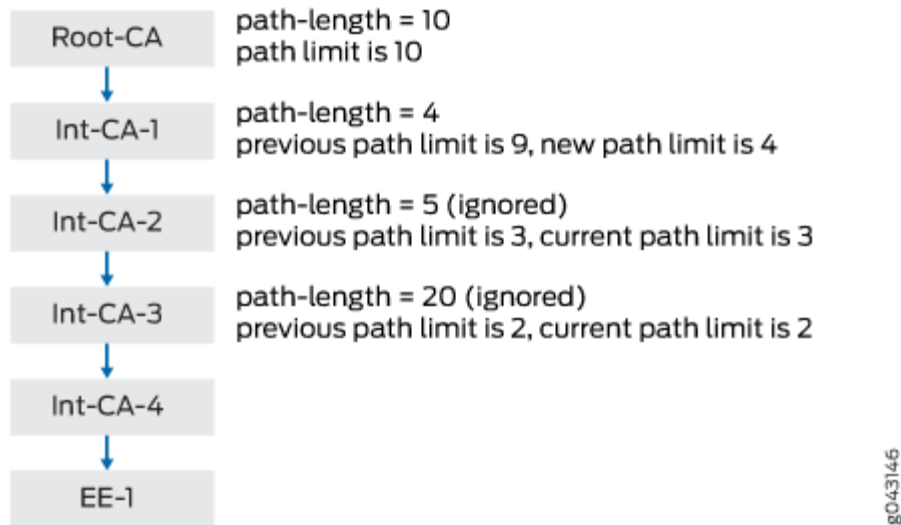
Certificate validation can involve a certificate chain that includes a root CA, one or more optional intermediate CAs, and an EE certificate. The number of intermediate CAs can grow depending upon the deployment scenario. Path length validation provides a mechanism to limit the number of intermediate certificates involved in certificate validation. **path-length** is an optional field in an X509 certificate. The value of **path-length** indicates the number of non-self-signed intermediate CA certificates allowed for certificate validation. The last certificate, which is generally the EE certificate, is not included in the path limit. If the root certificate contains a **path-length** value of 0, no intermediate CA certificates are allowed. If the **path-length** value is 1, there can be 0 or 1 intermediate CA certificates.

path-length can be present in multiple CA certificates in the certificate chain. The path length validation always begins with the self-signed root certificate. The path limit is decremented by 1 at each intermediate certificate in the chain. If an intermediate certificate contains a **path-length** value less than the current path limit, the new limit is enforced. On the other hand, if the **path-length** value is larger than the current path limit, it is ignored.

Figure 9 on page 60 shows a certificate chain that consists of a root CA, four intermediate CAs, and an EE. The **path-length** value in Root-CA is 10, therefore the initial path limit of non-self-signed intermediate CA certificates allowed for certificate validation is 10. At Int-CA-1, the path limit is 10-1 or 9. The **path-length** value in Int-CA-1 is 4, which is less than the path limit of 9, so the new path limit becomes 4. At Int-CA-2, the path limit is 4-1 or 3. The **path-length** value in Int-CA-2 is 5, which is larger

than the path limit of 3, so it is ignored. At Int-CA-3, the path limit is 3-1 or 2. The **path-length** value in Int-CA-3 is 20, which is larger than the path limit of 2, so it is also ignored.

Figure 9: Path Length Validation



Key Usage

The key usage field in an EE or CA certificate defines the purpose of the key contained in the certificate.

- For EE certificates, if the key usage field is present but the certificate does not contain **digitalSignature** or **nonrepudiation** flags, the certificate is rejected. If the key usage field is not present, then key usage is not checked.
- For CA certificates, the key can be used for certificate or CRL signature validation. Because the PKI daemon is responsible for both X509 certificate validation and CRL downloads, key usage must be checked before validating the certificate or CRL.

In certificate signature validation, the **keyCertSign** flag indicates that a CA certificate can be used for certificate signature validation. If this flag is not set, certificate validation is terminated.

In Phase 1 negotiations of CRL signature validation, participants check the certificate revocation list (CRL) to see if certificates received during an IKE exchange are still valid. The CRL is periodically downloaded for CA profiles configured with CRL as the certificate revocation check. Downloaded CRL files must be verified before they are downloaded into the device. One of the verification steps is to validate the CRL signature using a CA certificate. The downloaded CRL is signed with the CA certificate's private key and it must be verified with the CA certificate's public key stored in the

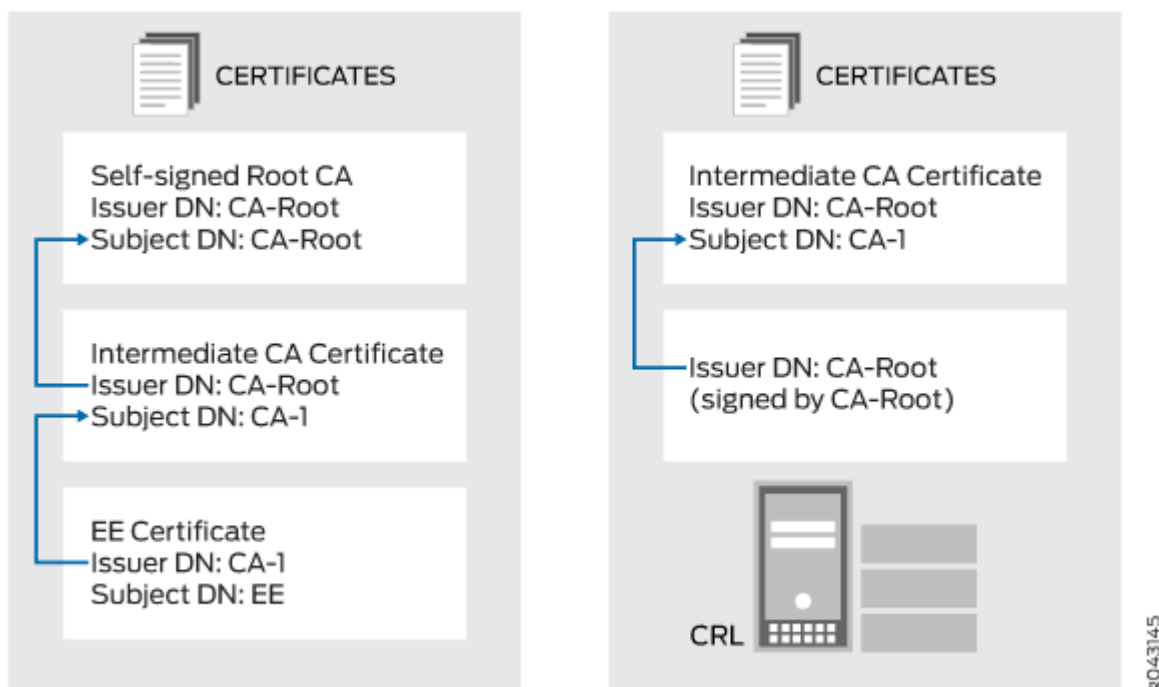
device. The key usage field in the CA certificate must contain the **CRLSign** flag to verify the downloaded CRL. If this flag is not present, the CRL is discarded.

Issuer and Subject Distinguished Name Validation

Signature validation is performed for certificates received from a peer as well as for the CRL file downloaded from a CA server. Signature validation involves looking up the CA certificate in a CA database based on the issuer's distinguished name (DN) in the certificate or the CRL being verified.

Figure 10 on page 61 shows the lookup for CA certificates based on the issuer DN. In the EE certificate, the issuer DN is CA-1, which is the subject DN of the intermediate CA certificate in the chain. In the intermediate CA certificate, the issuer DN is CA-Root, which is the subject DN of the self-signed Root-CA certificate in the chain. In the CRL, the issuer DN is CA-Root, which is the subject DN of the self-signed Root-CA certificate.

Figure 10: Issuer and Subject DN Validation



The lookup for the issuer or subject DN must follow these rules for attribute values:

- Attribute values encoded in different ASN.1 types (for example, PrintableString and BMPString) are assumed to represent different strings.

- Attribute values encoded in PrintableString types are not case-sensitive. These attribute values are compared after removing leading and trailing white spaces and converting internal substrings of one or more consecutive white spaces to a single space.
- Attribute values encoded in types other than PrintableString are case-sensitive.

Example: Validate Digital Certificate by Configuring Policy OIDs on an SRX Series Firewall

IN THIS SECTION

- [Requirements | 62](#)
- [Overview | 63](#)
- [Configuration | 63](#)
- [Verification | 64](#)

In some situations, it might be desirable to only accept certificates with known policy object identifiers (OIDs) from peers. This optional configuration allows certificate validation to succeed only if the certificate chain received from the peer contains at least one policy OID that is configured on the SRX Series Firewall. This example shows how to configure policy OIDs in the IKE policy on an SRX Series Firewall.

You must ensure that at least one of the policy OIDs configured on the SRX Series Firewall is included in a peer's certificate or certificate chain. Note that the **policy-oids** field in a peer's certificate is optional. If you configure policy OIDs in an IKE policy and the peer's certificate chain does not contain any policy OIDs, certificate validation for the peer fails.

Requirements

Before you begin:

- Ensure that you are using Junos OS Release 12.3X48-D10 or later for SRX Series Firewalls.
- Configure an IPsec VPN tunnel. See [IPsec VPN Configuration Overview](#). The complete IKE phase 1 and phase 2 VPN tunnel configuration is not shown in this example.

Overview

This example shows an IKE policy configuration where policy OIDs 2.16.840.1.101.3.1.48.2 and 5.16.40.1.101.3.1.55.2 are specified. The IKE policy `ike_cert_pol` references the IKE proposal `ike_cert_prop`, which is not shown. The local certificate on the SRX Series Firewall is `lc-igloo-root`.

Configuration

IN THIS SECTION

- [Procedure](#) | 63

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security ike policy ike_cert_pol mode main
set security ike policy ike_cert_pol proposals ike_cert_prop
set security ike policy ike_cert_pol certificate local-certificate lc-igloo-root
set security ike policy ike_cert_pol certificate policy-oids 2.16.840.1.101.3.1.48.2
set security ike policy ike_cert_pol certificate policy-oids 5.16.40.1.101.3.1.55.2
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see [Use the CLI Editor in Configuration Mode](#) in the [CLI User Guide](#).

To configure policy OIDs for certificate validation:

1. Configure the IKE policy:

```
[edit security ike policy ike_cert_pol]
user@host# set mode main
user@host# set proposals ike_cert_prop
```

```
user@host# set certificate local-certificate lc-igloo-root
user@host# set certificate policy-oids 2.16.840.1.101.3.1.48.2
user@host# set certificate policy-oids 5.16.40.1.101.3.1.55.2
```

Results

From configuration mode, confirm your configuration by entering the `show security ike policy ike_cert_pol` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show security ike policy ike_cert_pol
mode main;
proposals ike_cert_prop;
certificate {
    local-certificate lc-igloo-root;
    policy-oids [ 2.16.840.1.101.3.1.48.2 5.16.40.1.101.3.1.55.2 ];
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- [Verifying the CA Certificate | 64](#)
- [Verifying Policy OID Validation | 66](#)

Confirm that the configuration is working properly.

Verifying the CA Certificate

Purpose

Display the CA certificate configured on the device.

Action

From operational mode, enter the `show security pki ca-certificate ca-profile ca-tmp detail` command.

```
user@host> show security pki ca-certificate ca-profile ca-tmp detail
Certificate identifier: ca-tmp
Certificate version: 3
Serial number: 00000047
Issuer:
  Organization: U.S. Government,
  Organizational unit: DoD, Organizational unit: Testing, Country: US,
  Common name: Trust Anchor
Subject:
  Organization: U.S. Government,
  Organizational unit: Dod, Organizational unit: Testing, Country: US,
  Common name: CA1-PP.01.03
Subject string:
  C=US, O=U.S. Government, OU=Dod, OU=Testing, CN=CA1-PP.01.03
Validity:
  Not before: 01- 1-1998 12:01 UTC
  Not after: 01- 1-2048 12:01 UTC

?Public key algorithm: rsaEncryption(1024 bits)
30:81:89:02:81:81:00:cb:fd:78:0c:be:87:ac:cd:c0:33:66:a3:18
9e:fd:40:b7:9b:bc:dc:66:ff:08:45:f7:7e:fe:8e:d6:32:f8:5b:75
db:76:f0:4d:21:9a:6e:4f:04:21:4c:7e:08:a1:f9:3d:ac:8b:90:76
44:7b:c4:e9:9b:93:80:2a:64:83:6e:6a:cd:d8:d4:23:dd:ce:cb:3b
b5:ea:da:2b:40:8d:ad:a9:4d:97:58:cf:60:af:82:94:30:47:b7:7d
88:c3:76:c0:97:b4:6a:59:7e:f7:86:5d:d8:1f:af:fb:72:f1:b8:5c
2a:35:1e:a7:9e:14:51:d4:19:ae:c7:5c:65:ea:f5:02:03:01:00:01
Signature algorithm: sha1WithRSAEncryption
Certificate Policy:
  Policy Identifier = 2.16.840.1.101.3.1.48.2
Use for key: CRL signing, Certificate signing
Fingerprint:
  e0:b3:2f:2e:a1:c5:ee:ad:af:dd:96:85:f6:78:24:c5:89:ed:39:40 (sha1)
  f3:47:6e:55:bc:9d:80:39:5a:40:70:8b:10:0e:93:c5 (md5)
```


Verifying Policy OID Validation

Purpose

If the peer's certificate is successfully validated, IKE and IPsec security associations are established. If the validation of the peer's certificate fails, no IKE security association is established.

Action

From operational mode, enter the `show security ike security-associations` and `show security ipsec security-associations` commands.

```
user@host> show security ike security-associations
```

```
node0:
```

```
-----
Index   State Initiator cookie Responder cookie Mode      Remote Address
821765168 UP    88875c981252c1d8 b744ac9c21bde57e IKEv2     192.0.2.2
1106977837 UP    1a09e32d1e6f20f1 e008278091060acb IKEv2     198.51.100.202
```

```
user@host> show security ipsec security-associations
```

```
node0:
```

```
-----
Total active tunnels: 2
ID      Algorithm      SPI      Life:sec/kb Mon lsys Port Gateway
<213909506 ESP:aes-cbc-192/sha256 8cb9e40a 1295/ unlim - root 500 192.0.2.2
>213909506 ESP:aes-cbc-192/sha256 8271d2b2 1295/ unlim - root 500 192.0.2.2
<218365954 ESP:aes-cbc-192/sha256 d0153bc0 1726/ unlim - root 1495 198.51.100.202
>218365954 ESP:aes-cbc-192/sha256 97611813 1726/ unlim - root 1495 198.51.100.202
```

Meaning

The `show security ike security-associations` command lists all active IKE Phase 1 SAs. If no SAs are listed, there was a problem with Phase 1 establishment. In this case, check for the `PKID_CERT_POLICY_CHECK_FAIL` message in the system logs. This message indicates that the peer's certificate chain does not contain a policy OID that is configured on the SRX Series Firewall. Check the **policy-oids** values in the peer's certificate chain with the values configured on the SRX Series Firewall.

It might also be that the peer's certificate chain does not contain any **policy-oids** fields, which are optional fields. If this is the case, certificate validation fails if there are any policy OIDs configured on the SRX Series Firewall.

SEE ALSO

[Example: Configure a Certificate Authority Profile with CRL Locations](#) | 52

[Delete Certificates \(CLI Procedure\)](#) | 44

Dynamic Update of Trusted CA Certificates

SUMMARY

Read this topic to understand and configure dynamic update of default trusted CA certificates on your Junos OS devices.

IN THIS SECTION

- [Dynamic Update of Trusted CA Certificates](#) | 67
- [Configure Dynamic Update of Trusted CA Certificates](#) | 69

Dynamic Update of Trusted CA Certificates

IN THIS SECTION

- [Tasks involved in dynamic update of trusted CA bundle](#) | 68

The Junos OS device like an SRX Series Firewall provides a list of default trusted CA (Certificate Authority) certificates. These certificates are managed dynamically by the Junos OS device. You can also create a custom list of trusted CA certificates and load them into the device. But the custom trusted CA certificates needs to be managed manually. This section focuses on dynamic management of default trusted CA certificates.

With dynamic update of default trusted CA bundle -

- Removal of a CA in the event of compromise is taken care automatically.
- Addition of new CA to the default trusted CA bundle is immediate without having to wait for the new Junos OS release.

To load default trusted CA certificates, see [request security pki ca-certificate ca-profile-group load](#) with `filename default` option.

Tasks involved in dynamic update of trusted CA bundle

Following tasks are performed as part of dynamic update of default trusted CA bundle -

- Juniper CDN server (<http://signatures.juniper.net/cacert>) hosts the default trusted CA certificates.
- The server hosts signed copy of target file and manifest file along with the EE certificate to verify the signed copy of these files. The target file contains a list of default trusted CA certificates (default-trusted-ca-certs). The manifest file includes the revision number and date of modification of the default trusted CA bundle.
- Automatic download of trusted CA bundle is enabled by default in Junos OS device. You can either use default or non-default routing instance to connect to the Internet in order to download and update the default trusted CA certificates.
- Public Key Management (PKI) process using PKID securely downloads the default trusted CA bundle (default-trusted-ca-certs) from the CDN server into the device.

Dynamic update of trusted CA certificates does not manage any changes to the previously loaded ca-profile-group, manually added CA certificates and certificates that are part of other trusted groups.

See ["Configuring Dynamic Updated of Trusted CA Certificates" on page 69](#).

- Once the ca-profile-group load command is issued, PKI process loads the default trusted CA certificates in the background, unblocking the CLI, allowing you to proceed with other tasks.
- If there is no ca-profile-group associated with default-trusted-ca-certs, with each periodic polling, PKI still downloads the latest copy of trusted CA bundle to the device.
- If a CA certificate is deleted from the default trusted CA list, the PKI process ensures all references to the CA certificate are removed. If any references are present in the trusted-ca-group, it only holds the references to ca-profile names with actual CA certificates already deleted. See ["Configuring Dynamic Updated of Trusted CA Certificates" on page 69](#).
- PKI process periodically, by default every 24 hours, polls the CDN server for the latest default trusted CA bundle and updates the list for any changes to the trusted CAs in the bundle. If there are any changes, PKI process loads them in the background. You can optionally change the polling duration and also disable this auto-update process. See ["Configuring Dynamic Updated of Trusted CA Certificates" on page 69](#).

Configure Dynamic Update of Trusted CA Certificates

IN THIS SECTION

- [Check connectivity to the CDN server | 69](#)
- [Enable automatic download of default trusted CA certificates | 70](#)
- [Provide custom configuration for automatic download of default trusted CA certificates | 71](#)
- [Download default trusted CA certificates explicitly | 72](#)
- [Check the download status of default trusted CA certificates | 73](#)
- [Deactivate automatic download of trusted CA certificates | 74](#)

Prerequisites

Before configuring dynamic update of default trusted CA certificates, ensure to meet the following prerequisites -

- Basic configuration of Junos OS device is completed.
- Your Junos OS device is reachable to Juniper CDN server. You can use non-default routing instance as well to connect to Internet to download the default trusted CA certificates. Ensure non-default routing instance is configured prior to configuring dynamic update of trusted CA certificates. Contact Juniper sales for Juniper CDN server details.
- For custom CDN server, ensure to have latest CA certificates and the URL. Configuration of custom CDN server is out of scope of this topic.

Based on your requirements, navigate to the following tasks to configure dynamic update of default trusted CA bundle.

Check connectivity to the CDN server

IN THIS SECTION

- [Overview | 70](#)
- [Configuration | 70](#)

Overview

Use the following CLI to check connectivity to the CDN server for downloading default trusted CA certificates. This command downloads the manifest file and displays the trusted-ca-bundle version available in the CDN server.

See [request security pki ca-certificate ca-profile-group default-trusted-ca-certs](#), for details about the command.

Configuration

1. To check connectivity to the CDN server from operational mode of the Junos OS device -

```
user@host> request security pki ca-certificate ca-profile-group default-trusted-ca-certs
download check-server
```

Enable automatic download of default trusted CA certificates

IN THIS SECTION

- [Overview | 70](#)
- [Configuration | 71](#)

Overview

Juniper Networks regularly updates the default trusted CA certificates on Juniper CDN server and makes it available for download on Junos OS device. Automatic download of default trusted CA certificates is enabled by default on Junos OS device. You can customize the configuration and load the latest default trusted CA certificates at specified intervals. The default periodicity is 24 hours when you don't specify a value. When you use the default Juniper CDN Server (<http://signatures.juniper.net/cacert>), no separate configuration is needed.

This example shows how to enable automatic download of default trusted CA certificates on Junos OS device using default configuration settings. See [default-trusted-ca-certs \(Security\)](#) for details about the configuration statement. Loading of the downloaded default trusted CA certificates automatically happens in the background using the statement [request security pki ca-certificate ca-profile-group load](#) command. You don't have to explicitly run this command to load the certificates.

Configuration

As automatic download of default trusted CA certificates is enabled by default, no separate configuration is needed.

Provide custom configuration for automatic download of default trusted CA certificates

IN THIS SECTION

- [Overview | 71](#)
- [Configuration | 71](#)

Overview

In this example, you provide following custom configuration while enabling the automatic download of custom CA certificates -

- Configure the Junos OS device to download and install the default trusted CA certificates every 48 hours.
- Specify the custom CDN server reachable via the URL `signatures.example.net`.
- Specify non-default routing instance to reach the CDN server.

See [default-trusted-ca-certs \(Security\)](#) for details about the configuration statement.

Configuration

Configuration

1. Set the periodicity of download and load operations to 48 hours. This CLI automatically loads the certificates into the Junos OS device.

```
[edit]
user@host# set security pki default-trusted-ca-certs automatic-download interval hours 48
```

2. Specify the custom URL.

```
[edit]
user@host# set security pki default-trusted-ca-certs automatic-download url
signatures.example.net
```

3. Specify the routing instance.

```
[edit]
user@host# set security pki default-trusted-ca-certs automatic-download routing-instance RI1
```

4. Commit the configuration.

```
[edit]
user@host# commit
```

Download default trusted CA certificates explicitly

IN THIS SECTION

- [Overview | 72](#)
- [Configuration | 72](#)

Overview

Use the following CLI to manually download default trusted CA certificates to the Junos OS device from the CDN server. This command is in addition to automatic download of default trusted CA certs at regular intervals.

See [request security pki ca-certificate ca-profile-group default-trusted-ca-certs](#) for details about the command.

Configuration

Configuration

1. To explicitly download default trusted CA certificates from operational mode of the Junos OS device

```
user@host> request security pki ca-certificate ca-profile-group default-trusted-ca-certs  
download
```

Check the download status of default trusted CA certificates

IN THIS SECTION

- [Overview | 73](#)
- [Configuration | 73](#)

Overview

Use the following CLI to check the download status of default trusted CA certificates on the Junos OS device from the CDN server. This command displays the version number and version date. You can use this command to check the previous downloaded version and date.

See [request security pki ca-certificate ca-profile-group default-trusted-ca-certs](#) for details about the command.

Configuration

Configuration

1. To check the version number and version date available on the Junos OS device -

```
user@host> request security pki ca-certificate ca-profile-group default-trusted-ca-certs  
download status
```


Deactivate automatic download of trusted CA certificates

IN THIS SECTION

- [Overview | 74](#)
- [Configuration | 74](#)

Overview

Automatic download is enabled by default. This example shows how to deactivate the automatic download of default trusted CA certificates, though we don't recommend.

See [default-trusted-ca-certs \(Security\)](#) for details about the configuration statement.

Configuration

Configuration

1. To deactivate automatic download of default trusted CA certificates -

```
[edit]  
user@host# set security pki default-trusted-ca-certs automatic-download deactivate
```

2. Commit the configuration.

```
[edit]  
user@host# commit
```

ACME Protocol

SUMMARY

Learn about ACME protocol and how to enroll the certificate.

IN THIS SECTION

- [What is ACME Protocol | 75](#)
- [Enroll Local Certificate Using Let's Encrypt Server | 76](#)
- [Manual Re-Enroll Local Certificate | 78](#)
- [Delete ACME Account | 78](#)

What is ACME Protocol

IN THIS SECTION

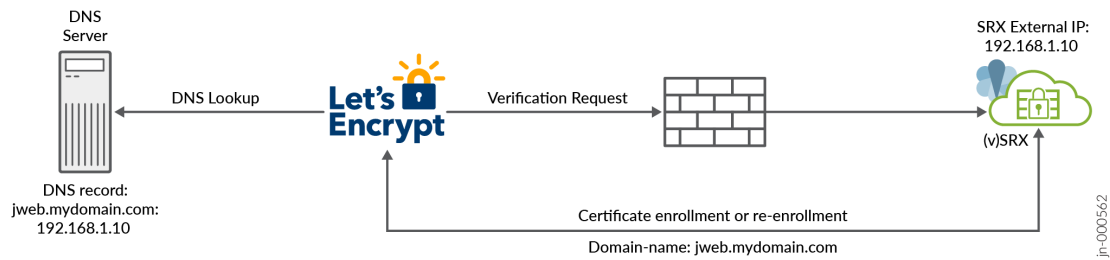
- [Limitations | 76](#)

Automated Certificate Management Environment (ACME) protocol is a new PKI enrollment standard used by several PKI servers such as Let's Encrypt. The Let's encrypt certificate allows for free usage of Web server certificates in SRX Series Firewalls, and this can be used in Juniper Secure Connect and J-Web. The Junos OS automatically re-enroll Let's Encrypt certificates on occurrence of every 25 days.

The ACME protocol allows the enrollment of certificates from Let's Encrypt server or ACME enabled servers. The SRX Series Firewalls enrolls the certificates from Let's Encrypt server and Juniper Secure Connect validates the certificates without copying and downloading any CA certificates.

When using Let's Encrypt, ensure that the Let's Encrypt server is able to resolve the domain name to the IP address of the SRX Series Firewall interface as shown in [Figure 11 on page 76](#). It must be able to reach the SRX Series Firewall interface on TCP port 80. During the certificate enrollment, the SRX Series Firewall will temporarily allow this incoming request automatically. If your SRX Series Firewall or an intermediate firewall or a router is blocking the TCP port 80, certificate enrollment will fail.

Figure 11: Name Resolution for Let's Encrypt



Limitations

- ACME specification - The dns-01 and external account binding are not supported.
- ACME cannot be used when J-Web listen to port 80
- Wildcard certificate is not supported such as *.mydomain.com, instead you can enroll multiple dns names.

Enroll Local Certificate Using Let's Encrypt Server

This example shows how to enroll the local certificate using Let's Encrypt.

1. Specify the CA profile.

```
[edit]

user@host#
set security pki ca-profile ISRG_Root_X1 ca-identity ISRG_Root_X1

user@host# set security pki ca-profile ISRG_Root_X1 revocation-check disable
user@host# set security pki ca-profile Lets_Encrypt ca-identity Lets_Encrypt

user@host#
set security pki ca-profile Lets_Encrypt enrollment url https://acme-v02.api.letsencrypt.org/
directory
```

2. Commit the configuration.

```
[edit]
user@host# commit
```

3. Load the CA certificate.

```
[edit]
user@host> request security pki ca-certificate load ca-profile ISRG_Root_X1 filename
ISRG_Root_X1.pem
```

4. Create ACME key ID.

```
[edit]
user@host> request security pki generate-key-pair size 2048 type rsa acme-key-id mydomain
```

5. Preparing enrollment of local certificate.

```
[edit]
user@host> request security pki generate-key-pair size 2048 type rsa certificate-id service-
mydomain
```

6. Enroll a certificate with one domain name.

```
[edit]
user@host> request security pki local-certificate enroll acme acme-key-id mydoamin
certificate-id service-mydomain ca-profile Lets_Encrypt domain-name jweb.mydomain.com email
jweb@acmejnpr.net letsencrypt-enrollment yes terms-of-service agree
```

Enroll a certificate with multiple domain names.

```
[edit]
user@host> request security pki local-certificate enroll acme acme-key-id mydomain
certificate-id service-mydomain ca-profile Lets_Encrypt domain-name jweb.mydomain.com,remote-
access.mydomain.com email jweb@acmejnpr.net letsencrypt-enrollment yes terms-of-service agree
```

7. Once the enrollment is finished the issued certificate will be loaded in certificate-id service-mydomain.

Manual Re-Enroll Local Certificate

To re-enroll a local certificate online:

1. Initiate the re-enrollment request.

```
[edit]
user@host> request security pki local-certificate re-enroll acme acme-key-id mydomain
certificate-id service-mydomain ca-profile Lets_Encrypt re-generate-keypair
```

2. Once the re-enrollment is finished the issued certificate will be loaded in certificate-id service-mydomain.

Delete ACME Account

To delete the ACME account:

1. Delete the ACME account.

```
[edit]
user@host> clear security pki acme account acme-key-id mydomain ca-profile Lets_Encrypt
```

You can delete the ACME account key only if the ACME is activated or created by the enrollment.

Configure Multiple Certificate Types to Establish IKE and IPsec SA

SUMMARY

Learn how to configure and manage multiple certificate types.

IN THIS SECTION

- [Requirements | 79](#)
- [Overview | 80](#)
- [Topology | 80](#)
- [Configuration | 81](#)
- [Verification | 91](#)

This example shows how to configure multiple certificate types to establish IKE and IPsec SA.

Starting in Junos OS Release 22.4R1, you can establish tunnels irrespective of the certificate type used on the initiator and responder if authentication-method is configured as certificates in IKE proposal using the `set security ike proposal ike_proposal_name authentication-method certificates` command.

You can view the certificate enrolled using `show security pki local-certificate certificate-id certificate-name detail` command.

You can verify the enrolled certificate using the `request security pki local-certificate verify certificate-id certificate-name` command.

Requirements

Before you begin:

- Ensure that you have certificates enrolled on your devices, see [Certificate Enrollment](#).

You can verify the certificates enrolled on your devices using the `request security pki local-certificate certificate-id certificate-name detail` command.

- Ensure that you have IKE package installed, to verify the installed IKE package use the `show version | match ike` operational command.

If you don't have the IKE package installed on the device, you can install the IKE package using the operational command `request system software add optional://junos-ike.tgz`, for more information, see [Enabling IPsec VPN Feature Set](#).

Overview

This example configures multiple certificate types to establish IKE and IPsec SA between on SRX_A and on SRX_B.

In this example, we have enrolled the RSA certificate on SRX_A and the ECDSA certificate on SRX_B devices. For more information about how to install the certificates, see [Certificate Enrollment](#).

Table 4: Topology Setup for SRX_A and SRX_B Devices

Device Name	Interface Used	IKE Gateway Address	IKE Gateway Local IP Address
SRX_A	ge-0/0/0	192.168.1.2	192.168.1.1
SRX_B	ge-0/0/0	192.168.1.1	192.168.1.2

Topology

The [Figure 12 on page 80](#) describes topology for multiple certificate types support configuration.

Figure 12: Multiple Certificate Types Support Configuration Example



Configuration

IN THIS SECTION

- [Configuring SRX_A | 81](#)
- [Configuring SRX_B | 86](#)

Configuring SRX_A

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/24
set interfaces ge-0/0/1 unit 0 family inet address 172.16.1.1/24
set interfaces st0 unit 1 family inet
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/1
set security zones security-zone untrust host-inbound-traffic system-services ike
set security zones security-zone untrust interfaces ge-0/0/0
set security zones security-zone VPN interfaces st0.1
set security policies from-zone VPN to-zone trust policy 1 match source-address any
set security policies from-zone VPN to-zone trust policy 1 match destination-address any
set security policies from-zone VPN to-zone trust policy 1 match application any
set security policies from-zone VPN to-zone trust policy 1 then permit
set security policies from-zone trust to-zone VPN policy 1 match source-address any
set security policies from-zone trust to-zone VPN policy 1 match destination-address any
set security policies from-zone trust to-zone VPN policy 1 match application any
set security policies from-zone trust to-zone VPN policy 1 then permit
set security policies default-policy deny-all
set security ike proposal IKE_PROP authentication-method certificates
set security ike proposal IKE_PROP dh-group group5
set security ike proposal IKE_PROP authentication-algorithm sha-256
set security ike proposal IKE_PROP encryption-algorithm aes-128-cbc
```



```

set security ike policy IKE_POL proposals IKE_PROP
set security ike policy IKE_POL certificate local-certificate r0_rsa_cert
set security ike gateway IKE_GW ike-policy IKE_POL
set security ike gateway IKE_GW address 192.168.1.2
set security ike gateway IKE_GW external-interface ge-0/0/0
set security ike gateway IKE_GW local-address 192.168.1.1
set security ike gateway IKE_GW version v2-only
set security ipsec proposal IPSEC_PROP protocol esp
set security ipsec proposal IPSEC_PROP authentication-algorithm hmac-sha-256-128
set security ipsec proposal IPSEC_PROP encryption-algorithm aes-192-cbc
set security ipsec policy IPSEC_POL proposals IPSEC_PROP
set security ipsec vpn IPSEC_VPN bind-interface st0.1
set security ipsec vpn IPSEC_VPN ike gateway IKE_GW
set security ipsec vpn IPSEC_VPN ike ipsec-policy IPSEC_POL
set security ipsec vpn IPSEC_VPN establish-tunnels on-traffic

```

Step-by-step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see [CLI Configuration Mode Overview](#) in the [CLI User Guide](#).

To configure multiple certificate types to establish IKE and IPsec SA:

1. View the certificates enrolled on your devices using the `show security pki local-certificate certificate-id certificate-name detail` command.

Install the certificate on your device if your device does not have the certificates enrolled. For more information, see [Certificate Enrollment](#).

2. Configure interfaces.

```

user@srxa# set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/24
user@srxa# set interfaces ge-0/0/1 unit 0 family inet address 172.16.1.1/24
user@srxa# set interfaces st0 unit 1 family inet

```

3. Configure security zones and the security policy.

```

user@srxa# set security zones security-zone trust host-inbound-traffic system-services all
user@srxa# set security zones security-zone trust host-inbound-traffic protocols all
user@srxa# set security zones security-zone trust interfaces ge-0/0/1
user@srxa# set security zones security-zone untrust host-inbound-traffic system-services ike

```

```

user@srxa# set security zones security-zone untrust interfaces ge-0/0/0
user@srxa# set security zones security-zone VPN interfaces st0.1
user@srxa# set security policies from-zone VPN to-zone trust policy 1 match source-address
any
user@srxa# set security policies from-zone VPN to-zone trust policy 1 match destination-
address any
user@srxa# set security policies from-zone VPN to-zone trust policy 1 match application any
user@srxa# set security policies from-zone VPN to-zone trust policy 1 then permit
user@srxa# set security policies from-zone trust to-zone VPN policy 1 match source-address
any
user@srxa# set security policies from-zone trust to-zone VPN policy 1 match destination-
address any
user@srxa# set security policies from-zone trust to-zone VPN policy 1 match application any
user@srxa# set security policies from-zone trust to-zone VPN policy 1 then permit
user@srxa# set security policies default-policy deny-all

```

4. Configure the IKE proposal.

```

[edit]
user@srxa# set security ike proposal IKE_PROP authentication-method certificates
user@srxa# set security ike proposal IKE_PROP dh-group group5
user@srxa# set security ike proposal IKE_PROP authentication-algorithm sha-256
user@srxa# set security ike proposal IKE_PROP encryption-algorithm aes-128-cbc

```

5. Configure the IKE policy.

```

[edit]
user@srxa# set security ike policy IKE_POL proposals IKE_PROP
user@srxa# set security ike policy IKE_POL certificate local-certificate r0_rsa.crt

```

6. Configure the IKE gateway.

```

[edit]
user@srxa# set security ike gateway IKE_GW ike-policy IKE_POL
user@srxa# set security ike gateway IKE_GW address 192.168.1.2
user@srxa# set security ike gateway IKE_GW external-interface ge-0/0/0
user@srxa# set security ike gateway IKE_GW local-address 192.168.1.1
user@srxa# set security ike gateway IKE_GW version v2-only

```

7. Configure the IPsec proposal.

```
[edit]
user@srxa# set security ipsec proposal IPSEC_PROP protocol esp
user@srxa# set security ipsec proposal IPSEC_PROP authentication-algorithm hmac-sha-256-128
user@srxa# set security ipsec proposal IPSEC_PROP encryption-algorithm aes-192-cbc
```

8. Configure the IPsec policy.

```
[edit]
user@srxa# set security ipsec policy IPSEC_POL proposals IPSEC_PROP
```

9. Configure the IPsec VPN.

```
[edit]
user@srxa# set security ipsec vpn IPSEC_VPN bind-interface st0.1
user@srxa# set security ipsec vpn IPSEC_VPN ike gateway IKE_GW
user@srxa# set security ipsec vpn IPSEC_VPN ike ipsec-policy IPSEC_POL
user@srxa# set security ipsec vpn IPSEC_VPN establish-tunnels on-traffic
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show security ike` and, `show security ipsec` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@srxa# show interfaces
ge-0/0/0 {
  description untrust;
  unit 0 {
    family inet {
      address 192.168.1.1/24;
    }
  }
}
ge-0/0/1 {
  description trust;
```

```

    unit 0 {
        family inet {
            address 172.16.1.1/24;
        }
    }
}
st0 {
    unit 1 {
        family inet;
    }
}

```

[edit]

user@srxa# **show security ike**

```

proposal IKE_PROP {
    authentication-method certificates;
    dh-group group5;
    authentication-algorithm sha-256;
    encryption-algorithm aes-128-cbc;
}
policy IKE_POL {
    proposals IKE_PROP;
    certificate {
        local-certificate r0_crt_rsa;
    }
}
gateway IKE_GW {
    ike-policy IKE_POL;
    address 192.168.1.2;
    external-interface ge-0/0/0;
    local-address 192.168.1.1;
    version v2-only;
}

```

[edit]

user@srxa# **show security ipsec**

```

proposal IPSEC_PROP {
    protocol esp;
    authentication-algorithm hmac-sha-256-128;
    encryption-algorithm aes-192-cbc;
}
policy IPSEC_POL {
    proposals IPSEC_PROP;
}

```

```

}
vpn IPSEC_VPN {
    bind-interface st0.1;
    ike {
        gateway IKE_GW;
        ipsec-policy IPSEC_POL;
    }
    establish-tunnels on-traffic;
}

```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring SRX_B

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```

set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24
set interfaces ge-0/0/1 unit 0 family inet address 172.18.1.2/24
set interfaces st0 unit 1 family inet
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/1
set security zones security-zone untrust host-inbound-traffic system-services ike
set security zones security-zone untrust interfaces ge-0/0/0
set security zones security-zone VPN interfaces st0.1
set security policies from-zone VPN to-zone trust policy 1 match source-address any
set security policies from-zone VPN to-zone trust policy 1 match destination-address any
set security policies from-zone VPN to-zone trust policy 1 match application any
set security policies from-zone VPN to-zone trust policy 1 then permit
set security policies from-zone trust to-zone VPN policy 1 match source-address any
set security policies from-zone trust to-zone VPN policy 1 match destination-address any
set security policies from-zone trust to-zone VPN policy 1 match application any
set security policies from-zone trust to-zone VPN policy 1 then permit
set security policies default-policy deny-all
set security ike proposal IKE_PROP authentication-method certificates
set security ike proposal IKE_PROP dh-group group5
set security ike proposal IKE_PROP authentication-algorithm sha-256

```

```

set security ike proposal IKE_PROP encryption-algorithm aes-128-cbc
set security ike policy IKE_POL proposals IKE_PROP
set security ike policy IKE_POL certificate local-certificate r1_cert_ecdsa384
set security ike gateway IKE_GW ike-policy IKE_POL
set security ike gateway IKE_GW address 192.168.1.1
set security ike gateway IKE_GW external-interface ge-0/0/0
set security ike gateway IKE_GW local-address 192.168.1.2
set security ike gateway IKE_GW version v2-only
set security ipsec proposal IPSEC_PROP protocol esp
set security ipsec proposal IPSEC_PROP authentication-algorithm hmac-sha-256-128
set security ipsec proposal IPSEC_PROP encryption-algorithm aes-192-cbc
set security ipsec policy IPSEC_POL proposals IPSEC_PROP
set security ipsec vpn IPSEC_VPN bind-interface st0.1
set security ipsec vpn IPSEC_VPN ike gateway IKE_GW
set security ipsec vpn IPSEC_VPN ike ipsec-policy IPSEC_POL
set security ipsec vpn IPSEC_VPN establish-tunnels on-traffic

```

Step-by-step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see [CLI Configuration Mode Overview](#) in the [CLI User Guide](#).

To configure multiple certificate types to establish IKE and IPsec SA:

1. View the certificates enrolled on your devices using the `request security pki local-certificate certificate-id certificate-name detail` command.

Install the certificate on your device if your device does not have the certificates enrolled. For more information, see [Certificate Enrollment](#).

2. Configure interfaces.

```

user@srxb# set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24
user@srxb# set interfaces ge-0/0/1 unit 0 family inet address 172.18.1.2/24
user@srxb# set interfaces st0 unit 1 family inet

```

3. Configure security zones and the security policy.

```

user@srxb# set security zones security-zone trust host-inbound-traffic system-services all
user@srxb# set security zones security-zone trust host-inbound-traffic protocols all
user@srxb# set security zones security-zone trust interfaces ge-0/0/1

```

```

user@srxb# set security zones security-zone untrust host-inbound-traffic system-services ike
user@srxb# set security zones security-zone untrust interfaces ge-0/0/0
user@srxb# set security zones security-zone VPN interfaces st0.1
user@srxb# set security policies from-zone VPN to-zone trust policy 1 match source-address
any
user@srxb# set security policies from-zone VPN to-zone trust policy 1 match destination-
address any
user@srxb# set security policies from-zone VPN to-zone trust policy 1 match application any
user@srxb# set security policies from-zone VPN to-zone trust policy 1 then permit
user@srxb# set security policies from-zone trust to-zone VPN policy 1 match source-address
any
user@srxb# set security policies from-zone trust to-zone VPN policy 1 match destination-
address any
user@srxb# set security policies from-zone trust to-zone VPN policy 1 match application any
user@srxb# set security policies from-zone trust to-zone VPN policy 1 then permit
user@srxb# set security policies default-policy deny-all

```

4. Configure the IKE proposal.

```

[edit]
user@srxb# set security ike proposal IKE_PROP authentication-method certificates
user@srxb# set security ike proposal IKE_PROP dh-group group5
user@srxb# set security ike proposal IKE_PROP authentication-algorithm sha-256
user@srxb# set security ike proposal IKE_PROP encryption-algorithm aes-128-cbc

```

5. Configure the IKE policy.

```

[edit]
user@srxb# set security ike policy IKE_POL proposals IKE_PROP
user@srxb# set security ike policy IKE_POL certificate local-certificate r1_cert_ecdsa384

```

6. Configure the IKE gateway.

```

[edit]
user@srxb# set security ike gateway IKE_GW ike-policy IKE_POL
user@srxb# set security ike gateway IKE_GW address 192.168.1.1
user@srxb# set security ike gateway IKE_GW external-interface ge-0/0/0
user@srxb# set security ike gateway IKE_GW local-address 192.168.1.2
user@srxb# set security ike gateway IKE_GW version v2-only

```

7. Configure the IPsec proposal.

```
[edit]
user@srxb# set security ipsec proposal IPSEC_PROP protocol esp
user@srxb# set security ipsec proposal IPSEC_PROP authentication-algorithm hmac-sha-256-128
user@srxb# set security ipsec proposal IPSEC_PROP encryption-algorithm aes-192-cbc
```

8. Configure the IPsec policy.

```
[edit]
user@srxb# set security ipsec policy IPSEC_POL proposals IPSEC_PROP
```

9. Configure the IPsec VPN.

```
[edit]
user@srxb# set security ipsec vpn IPSEC_VPN bind-interface st0.1
user@srxb# set security ipsec vpn IPSEC_VPN ike gateway IKE_GW
user@srxb# set security ipsec vpn IPSEC_VPN ike ipsec-policy IPSEC_POL
user@srxb# set security ipsec vpn IPSEC_VPN establish-tunnels immediately
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show security ike` and, `show security ipsec` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@srxb# show interfaces
ge-0/0/0 {
  description untrust;
  unit 0 {
    family inet {
      address 192.168.1.2/24;
    }
  }
}
ge-0/0/1 {
  description trust;
```



```

    unit 0 {
        family inet {
            address 172.18.1.2/24;
        }
    }
}
st0 {
    unit 1 {
        family inet;
    }
}

```

[edit]

user@srxb# **show security ike**

```

proposal IKE_PROP {
    authentication-method certificates;
    dh-group group5;
    authentication-algorithm sha-256;
    encryption-algorithm aes-128-cbc;
}
policy IKE_POL {
    proposals IKE_PROP;
    certificate {
        local-certificate r1.crt_ecdsa384;
    }
}
gateway IKE_GW {
    ike-policy IKE_POL;
    address 192.168.1.1;
    external-interface ge-0/0/0;
    local-address 192.168.1.2;
    version v2-only;
}

```

[edit]

user@srxb# **show security ipsec**

```

proposal IPSEC_PROP {
    protocol esp;
    authentication-algorithm hmac-sha-256-128;
    encryption-algorithm aes-192-cbc;
}
policy IPSEC_POL {
    proposals IPSEC_PROP;
}

```

```
}  
vpn IPSEC_VPN {  
    bind-interface st0.1;  
    ike {  
        gateway IKE_GW;  
        ipsec-policy IPSEC_POL;  
    }  
    establish-tunnels immediately;  
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- [Verify SRX_A | 91](#)
- [Verify SRX_B | 97](#)

Confirm that the configuration is working properly.

Verify SRX_A

The sample outputs shown are on SRX-A.

Purpose

Verify the IPsec Phase 2 status.

Action

From operational mode, enter the `show security ike security-associations` command.

```
user@srxa> show security ike security-associations
```

Index	State	Initiator cookie	Responder cookie	Mode	Remote Address
32	UP	6723643250f0f357	f6295f11b0d7c8ab	IKEv2	192.168.1.2

From operational mode, enter the `show security ipsec security-associations` command.

```
user@srxa> show security ipsec security-associations
```

ID	Algorithm	SPI	Life:sec/kb	Mon lsys	Port	Gateway
<500033	ESP:aes-cbc-192/sha256	0x5f156c1b	2750/	unlim	- root 500	192.168.1.2
>500033	ESP:aes-cbc-192/sha256	0x7ea065e7	2750/	unlim	- root 500	192.168.1.2

From operational mode, enter the `show security ike security-associations detail` command.

```
user@srxa> show security ike security-associations detail
```

IKE peer 192.168.1.2, Index 32, Gateway Name: IKE_GW
 Role: Responder, State: UP
 Initiator cookie: 6723643250f0f357, Responder cookie: f6295f11b0d7c8ab
 Exchange type: IKEv2, Authentication method: RSA-signatures
 Local gateway interface: ge-0/0/0.0
 Routing instance: default
 Local: 192.168.1.1:500, Remote: 192.168.1.2:500
 Lifetime: Expires in 28165 seconds
 Reauth Lifetime: Disabled
 IKE Fragmentation: Enabled, Size: 576
 Remote Access Client Info: Unknown Client
 Peer ike-id: 192.168.1.2
 AAA assigned IP: 0.0.0.0
 Algorithms:
 Authentication : hmac-sha256-128
 Encryption : aes128-cbc
 Pseudo random function: hmac-sha256
 Diffie-Hellman group : DH-group-5
 Traffic statistics:
 Input bytes : 1346
 Output bytes : 1887

```

Input  packets:          3
Output packets:          4
Input  fragmented packets: 2
Output fragmented packets: 3
IPSec security associations: 2 created, 0 deleted
Phase 2 negotiations in progress: 1
IPSec Tunnel IDs: 500033

Negotiation type: Quick mode, Role: Responder, Message ID: 0
Local: 192.168.1.1:500, Remote: 192.168.1.2:500
Local identity: 192.168.1.1
Remote identity: 192.168.1.2
Flags: IKE SA is created

```

IPsec SA Rekey CREATE_CHILD_SA exchange stats:

Initiator stats:		Responder stats:	
Request Out	: 0	Request In	:
0			
Response In	: 0	Response Out	:
0			
No Proposal Chosen In	: 0	No Proposal Chosen Out	:
0			
Invalid KE In	: 0	Invalid KE Out	:
0			
TS Unacceptable In	: 0	TS Unacceptable Out	:
0			
Res DH Compute Key Fail	: 0	Res DH Compute Key Fail:	
0			
Res Verify SA Fail	: 0		
Res Verify DH Group Fail	: 0		
Res Verify TS Fail	: 0		

From operational mode, enter the `show security ipsec security-associations detail` command.

```

user@srxa> show security ipsec security-associations detail
ID: 500033 Virtual-system: root, VPN Name: IPSEC_VPN
Local Gateway: 192.168.1.1, Remote Gateway: 192.168.1.2
Local Identity: ipv4(0.0.0.0-255.255.255.255)
Remote Identity: ipv4(0.0.0.0-255.255.255.255)
TS Type: proxy-id
Version: IKEv2
PFS group: N/A

```

```

DF-bit: clear, Copy-Outer-DSCP Disabled, Bind-interface: st0.1, Tunnel MTU: 0, Policy-name:
IPSEC_POL
Port: 500, Nego#: 0, Fail#: 0, Def-Del#: 0 Flag: 0
Multi-sa, Configured SAs# 0, Negotiated SAs#: 0
Tunnel events:
  Thu Mar 09 2023 22:41:36: IPsec SA negotiation succeeds (1 times)
Location: FPC 0, PIC 0, KMD-Instance 0
Anchorship: Thread 1
Distribution-Profile: default-profile
Direction: inbound, SPI: 0x5f156c1b, AUX-SPI: 0
              , VPN Monitoring: -
  Hard lifetime: Expires in 2895 seconds
  Lifesize Remaining: Unlimited
  Soft lifetime: Expires in 2286 seconds
  Mode: Tunnel(0 0), Type: dynamic, State: installed
  Protocol: ESP, Authentication: hmac-sha256-128, Encryption: aes-cbc (192 bits)
  Anti-replay service: counter-based enabled, Replay window size: 64
  Extended-Sequence-Number: Disabled
  tunnel-establishment: establish-tunnels-on-traffic
  IKE SA Index: 32
Direction: outbound, SPI: 0x7ea065e7, AUX-SPI: 0
              , VPN Monitoring: -
  Hard lifetime: Expires in 2895 seconds
  Lifesize Remaining: Unlimited
  Soft lifetime: Expires in 2286 seconds
  Mode: Tunnel(0 0), Type: dynamic, State: installed
  Protocol: ESP, Authentication: hmac-sha256-128, Encryption: aes-cbc (192 bits)
  Anti-replay service: counter-based enabled, Replay window size: 64
  Extended-Sequence-Number: Disabled
  tunnel-establishment: establish-tunnels-on-traffic
  IKE SA Index: 32

```

From operational mode, enter the `show security pki local-certificate certificate-id r0_rsa_cr detail` command.

```

user@srxa> show security pki local-certificate certificate-id r0_rsa_cr detail
LSYS: root-logical-system
Certificate identifier: r0_rsa_cr
Certificate version: 3

Serial number:
  hexadecimal: 0x0186a62478ae8f0cdd766eb38dbd53

```

```

decimal: 7923302907757301847007106226306387
Issuer:
  Organization: juniper, Country: India, Common name: Root-CA
Subject:
  Organization: juniper, Organizational unit: marketing, State: california, Locality:
sunnyvale, Common name: r0, Domain component: juniper
Subject string:
  DC=juniper, CN=r0, OU=marketing, O=juniper, L=sunnyvale, ST=california, C=us
Alternate subject: "r0@juniper.net", r0.juniper.net, 192.168.1.1
Cert-Chain: Root-CA
Validity:
  Not before: 03- 3-2023 05:54 UTC
  Not after: 06- 6-2027 12:36 UTC
Public key algorithm: rsaEncryption(2048 bits)
  30:82:01:0a:02:82:01:01:00:b0:e5:53:8d:7e:20:fa:6b:21:c2:d1
  2b:48:8f:af:c3:eb:8b:23:4a:f7:c5:1f:cf:2c:6a:b3:2e:8a:ef:1b
  f7:97:aa:fd:1d:ab:1c:76:9b:40:a3:ac:bb:49:f6:93:f9:e1:4e:62
  df:3d:ca:e5:d2:95:9c:a0:f4:2b:d7:7e:1d:20:94:69:a8:e4:cf:dc
  15:90:4c:be:1d:d8:1c:52:08:3a:d1:05:a3:bb:2f:8f:31:0c:6b:21
  ef:76:c3:c7:fb:be:4a:cb:da:cc:8d:04:3a:75:0c:eb:5d:e2:f6:13
  50:fe:39:67:c0:77:2f:32:b0:5e:38:6f:9c:79:b3:5d:f3:57:f4:f8
  42:f5:22:5b:6c:58:67:90:4e:1e:ec:6a:03:e2:c0:87:65:02:ca:da
  6f:95:0a:8c:2a:fd:45:4f:3a:b5:ef:18:05:1c:54:e6:fe:45:bb:73
  53:81:b2:c6:b7:36:36:57:6d:9c:d3:d9:80:e7:d6:85:92:74:32:88
  16:01:03:27:57:76:8e:5e:d6:73:ac:bf:68:fd:6d:a1:2a:8f:f5:3a
  29:b0:c9:44:9b:c8:46:c1:bf:c0:52:2a:f0:51:be:b5:f6:e1:f5:3e
  96:1d:3a:42:29:28:d3:cf:60:b9:eb:24:04:47:d3:f1:3f:5e:38:fc
  7f:33:f6:94:9d:02:03:01:00:01
Signature algorithm: sha256WithRSAEncryption
Fingerprint:
  4d:f6:89:c5:d6:3c:74:73:db:3e:f6:4b:1e:26:6c:c1:1c:1d:a7:4d (sha1)
  6b:1c:a8:1f:de:5a:9b:3e:d5:c4:85:29:af:3f:82:f2 (md5)

6b:7a:b5:d1:57:cf:75:9d:1f:63:b9:f6:49:e4:4e:b3:13:2c:83:f1:f7:25:44:6f:45:2f:0d:2f:ae:a8:80:85
(sh256)
Auto-re-enrollment:
  Status: Disabled
  Next trigger time: Timer not started

```

From operational mode, enter the `show security pki ca-certificate ca-profile Root-CA detail` command.

```

user@srxa> show security pki ca-certificate ca-profile Root-CA detail
  LSYS: root-logical-system
  CA profile: Root-CA
Certificate identifier: Root-CA
  Certificate version: 3

Serial number:
  hexadecimal: 0x00000440
  decimal: 1088
Issuer:
  Organization: juniper, Country: India, Common name: Root-CA
Subject:
  Organization: juniper, Country: India, Common name: Root-CA
Subject string:
  C=India, O=juniper, CN=Root-CA
Validity:
  Not before: 06- 7-2022 12:36 UTC
  Not after: 06- 6-2027 12:36 UTC
Public key algorithm: rsaEncryption(2048 bits)
  30:82:01:0a:02:82:01:01:00:cd:9c:e6:9f:62:6c:49:15:c2:da:eb
  8e:e6:e5:a1:88:40:d8:b5:2e:5b:1a:0e:de:96:d7:0b:19:f9:03:44
  98:49:d5:cc:a8:90:2b:7f:1b:58:7b:1f:26:92:18:4c:2d:37:65:5c
  9f:0f:6e:10:b5:34:6f:2d:b5:9c:27:3b:a6:b1:b5:a0:e2:a6:92:3d
  e4:68:fe:5d:71:06:6f:ce:e6:0f:0f:e3:94:2a:23:57:98:a0:6a:9c
  e0:52:a2:47:ff:ce:b0:47:bd:36:95:80:a7:af:d2:49:b1:5d:2a:3d
  28:e4:95:06:b8:b3:d9:07:11:3c:13:af:c6:e2:51:08:22:82:2d:ec
  4f:26:40:b0:b0:55:2d:6e:c0:c8:19:34:a7:99:5a:bc:58:98:69:ae
  04:d6:6d:ec:4a:c9:55:a5:ff:00:cb:3b:02:85:fa:02:a1:5c:c1:9d
  6d:44:b8:95:8f:77:c0:53:fc:7f:a4:09:a3:25:1c:4a:e2:9d:0c:81
  08:b4:c8:b8:0d:bc:94:75:54:75:57:4f:d3:a4:17:0d:5d:1a:f3:c1
  1d:5d:73:2f:fe:8b:cb:fc:1f:93:87:72:d6:be:df:86:d7:e6:d1:c7
  0d:00:1a:6e:58:db:6a:1c:2f:1d:17:46:9a:f2:69:b4:21:db:08:5d
  8d:ab:30:7d:7f:02:03:01:00:01
Signature algorithm: sha256WithRSAEncryption
Distribution CRL:
  http://10.102.40.55:8080/crl-as-der/currentcrl-11.crl?id=11
Use for key: CRL signing, Certificate signing, Key encipherment, Digital signature
Fingerprint:
  8b:84:60:2a:58:5b:80:f0:b9:ae:25:9f:67:3d:d6:81:ee:43:6c:d4 (sha1)
  ab:ec:4d:fe:d4:04:9c:c9:79:1d:9a:33:4e:6d:78:f6 (md5)

```

```
9d:f0:c0:a0:93:74:11:53:d3:4d:2d:75:d3:60:37:5f:fb:b7:a9:67:42:cd:7c:3c:0e:0f:9b:58:36:3c:14:f5
(sh256)
```

Verify SRX_B

The sample outputs shown are on SRX-B.

Purpose

Verify the IPsec Phase 2 status.

Action

From operational mode, enter the `show security ike security-associations` command.

```
user@srxb> show security ike security-associations
Index   State Initiator cookie Responder cookie Mode      Remote Address
56042   UP    6723643250f0f357 f6295f11b0d7c8ab IKEv2     192.168.1.1
```

From operational mode, enter the `show security ipsec security-associations` command.

```
user@srxb> show security ipsec security-associations
Total active tunnels: 1      Total IPsec sas: 1
ID      Algorithm      SPI      Life:sec/kb Mon lsys Port Gateway
<500230 ESP:aes-cbc-192/sha256 0x7ea065e7 2638/ unlim - root 500 192.168.1.1
>500230 ESP:aes-cbc-192/sha256 0x5f156c1b 2638/ unlim - root 500 192.168.1.1
```

From operational mode, enter the `show security ike security-associations detail` command.

```
user@srxb> show security ike security-associations detail
IKE peer 192.168.1.1, Index 56042, Gateway Name: IKE_GW
Role: Responder, State: UP
Initiator cookie: 6723643250f0f357, Responder cookie: f6295f11b0d7c8ab
Exchange type: IKEv2, Authentication method: ECDSA-384-signatures
Local gateway interface: ge-0/0/0.0
Routing instance: default
Local: 192.168.1.2:500, Remote: 192.168.1.1:500
Lifetime: Expires in 18995 seconds
```



```

Reauth Lifetime: Disabled
IKE Fragmentation: Enabled, Size: 576
Remote Access Client Info: Unknown Client
Peer ike-id: 192.168.1.1
AAA assigned IP: 0.0.0.0
Algorithms:
  Authentication      : hmac-sha256-128
  Encryption          : aes128-cbc
  Pseudo random function: hmac-sha256
  Diffie-Hellman group : DH-group-5
Traffic statistics:
  Input bytes  :          2934
  Output bytes :          2379
  Input packets:           10
  Output packets:           9
  Input fragmented packets: 3
  Output fragmented packets: 2
IPSec security associations: 8 created, 3 deleted
Phase 2 negotiations in progress: 1
IPSec Tunnel IDs: 500230

```

```

Negotiation type: Quick mode, Role: Responder, Message ID: 0
Local: 192.168.1.2:500, Remote: 192.168.1.1:500
Local identity: 192.168.1.2
Remote identity: 192.168.1.1
Flags: IKE SA is created

```

```
IPsec SA Rekey CREATE_CHILD_SA exchange stats:
```

Initiator stats:		Responder stats:	
Request Out	: 1	Request In	:
Response In	: 1	Response Out	:
No Proposal Chosen In	: 0	No Proposal Chosen Out	:
Invalid KE In	: 0	Invalid KE Out	:
TS Unacceptable In	: 0	TS Unacceptable Out	:
Res DH Compute Key Fail	: 0	Res DH Compute Key Fail:	
Res Verify SA Fail	: 0		

```
Res Verify DH Group Fail: 0
Res Verify TS Fail      : 0
```

From operational mode, enter the `show security ipsec security-associations detail` command.

```
user@srxb> show security ipsec security-associations detail
ID: 500230 Virtual-system: root, VPN Name: IPSEC_VPN
Local Gateway: 192.168.1.2, Remote Gateway: 192.168.1.1
Local Identity: ipv4(0.0.0.0-255.255.255.255)
Remote Identity: ipv4(0.0.0.0-255.255.255.255)
TS Type: proxy-id
Version: IKEv2
PFS group: N/A
DF-bit: clear, Copy-Outer-DSCP Disabled, Bind-interface: st0.1, Tunnel MTU: 0, Policy-name:
IPSEC_POL
Port: 500, Nego#: 0, Fail#: 0, Def-Del#: 0 Flag: 0
Multi-sa, Configured SAs# 0, Negotiated SAs#: 0
Tunnel events:
  Thu Mar 02 2023 22:26:16: IPsec SA negotiation succeeds (1 times)
Location: FPC 0, PIC 0, KMD-Instance 0
Anchorship: Thread 1
Distribution-Profile: default-profile
Direction: inbound, SPI: 0x7ea065e7, AUX-SPI: 0
              , VPN Monitoring: -
  Hard lifetime: Expires in 2633 seconds
  Lifesize Remaining: Unlimited
  Soft lifetime: Expires in 2002 seconds
  Mode: Tunnel(0 0), Type: dynamic, State: installed
  Protocol: ESP, Authentication: hmac-sha256-128, Encryption: aes-cbc (192 bits)
  Anti-replay service: counter-based enabled, Replay window size: 64
  Extended-Sequence-Number: Disabled
  tunnel-establishment: establish-tunnels-on-traffic
  IKE SA Index: 56042
Direction: outbound, SPI: 0x5f156c1b, AUX-SPI: 0
              , VPN Monitoring: -
  Hard lifetime: Expires in 2633 seconds
  Lifesize Remaining: Unlimited
  Soft lifetime: Expires in 2002 seconds
  Mode: Tunnel(0 0), Type: dynamic, State: installed
  Protocol: ESP, Authentication: hmac-sha256-128, Encryption: aes-cbc (192 bits)
  Anti-replay service: counter-based enabled, Replay window size: 64
  Extended-Sequence-Number: Disabled
```

```
tunnel-establishment: establish-tunnels-on-traffic
IKE SA Index: 56042
```

From operational mode, enter the `show security pki local-certificate certificate-id r1.crt_ecdsa384 detail` command.

```
user@srxb> show security pki local-certificate certificate-id r1.crt_ecdsa384 detail
  LSYS: root-logical-system
Certificate identifier: r1.crt_ecdsa384
Certificate version: 3

Serial number:
  hexadecimal: 0x0186a6254347a38063946d08595a55
  decimal: 7923303152683216740296668848151125
Issuer:
  Organization: juniper, Country: India, Common name: root-ecdsa-384
Subject:
  Organization: juniper, Organizational unit: marketing, State: california, Locality:
sunnyvale, Common name: r1_spk1, Domain component: juniper
Subject string:
  DC=juniper, CN=r1_spk1, OU=marketing, O=juniper, L=sunnyvale, ST=california, C=us
Alternate subject: "r1_spk1@juniper.net", r1_spk1.juniper.net, 192.168.2
Cert-Chain: root-ecdsa-384
Validity:
  Not before: 03- 3-2023 05:55 UTC
  Not after: 06- 6-2027 13:21 UTC
Public key algorithm: ecdsaEncryption(384 bits)
  04:c2:ba:19:dc:0d:62:a7:94:7b:9b:1d:4d:ff:a1:e1:44:b5:57:a7
  cb:7d:33:6b:35:87:b8:e4:ca:44:b1:6c:6d:63:ae:6f:3c:31:7c:7e
  65:99:b3:2d:a3:76:30:23:e5:0e:34:e1:28:54:d6:3e:d3:8b:de:b6
  b9:45:05:82:6f:1d:20:b7:6f:3c:ce:a2:13:a2:b4:37:0b:db:35:1e
  20:54:b5:06:9d:f8:7f:19:7b:c5:d7:7b:57:8b:28:31:d3
Signature algorithm: ecdsa-with-SHA384
Fingerprint:
  9b:cb:5a:57:a8:60:a0:ee:5c:be:59:4c:db:35:39:d3:b7:29:ef:b1 (sha1)
  ef:b5:e3:be:35:1b:6e:02:0b:61:11:a5:53:07:b4:89 (md5)

8f:86:d0:12:ea:bc:a8:81:a8:17:3a:f9:03:e4:91:57:20:9c:11:bc:a4:dd:d1:7f:d1:48:3f:5b:d9:fb:93:32
(sh256)
Auto-re-enrollment:
```

```
Status: Disabled
Next trigger time: Timer not started
```

s

From operational mode, enter the `show security pki ca-certificate ca-profile Root-CA detail` command.

```
user@srxb> show security pki ca-certificate ca-profile Root-CA detail
LSYS: root-logical-system
CA profile: Root-CA
Certificate identifier: Root-CA
Certificate version: 3

Serial number:
  hexadecimal: 0x00000440
  decimal: 1088
Issuer:
  Organization: juniper, Country: India, Common name: Root-CA
Subject:
  Organization: juniper, Country: India, Common name: Root-CA
Subject string:
  C=India, O=juniper, CN=Root-CA
Validity:
  Not before: 06- 7-2022 12:36 UTC
  Not after: 06- 6-2027 12:36 UTC
Public key algorithm: rsaEncryption(2048 bits)
30:82:01:0a:02:82:01:01:00:cd:9c:e6:9f:62:6c:49:15:c2:da:eb
8e:e6:e5:a1:88:40:d8:b5:2e:5b:1a:0e:de:96:d7:0b:19:f9:03:44
98:49:d5:cc:a8:90:2b:7f:1b:58:7b:1f:26:92:18:4c:2d:37:65:5c
9f:0f:6e:10:b5:34:6f:2d:b5:9c:27:3b:a6:b1:b5:a0:e2:a6:92:3d
e4:68:fe:5d:71:06:6f:ce:e6:0f:0f:e3:94:2a:23:57:98:a0:6a:9c
e0:52:a2:47:ff:ce:b0:47:bd:36:95:80:a7:af:d2:49:b1:5d:2a:3d
28:e4:95:06:b8:b3:d9:07:11:3c:13:af:c6:e2:51:08:22:82:2d:ec
4f:26:40:b0:b0:55:2d:6e:c0:c8:19:34:a7:99:5a:bc:58:98:69:ae
04:d6:6d:ec:4a:c9:55:a5:ff:00:cb:3b:02:85:fa:02:a1:5c:c1:9d
6d:44:b8:95:8f:77:c0:53:fc:7f:a4:09:a3:25:1c:4a:e2:9d:0c:81
08:b4:c8:b8:0d:bc:94:75:54:75:57:4f:d3:a4:17:0d:5d:1a:f3:c1
1d:5d:73:2f:fe:8b:cb:fc:1f:93:87:72:d6:be:df:86:d7:e6:d1:c7
0d:00:1a:6e:58:db:6a:1c:2f:1d:17:46:9a:f2:69:b4:21:db:08:5d
8d:ab:30:7d:7f:02:03:01:00:01
Signature algorithm: sha256WithRSAEncryption
Distribution CRL:
```

```
http://10.102.40.55:8080/crl-as-der/currentcrl-11.crl?id=11
```

```
Use for key: CRL signing, Certificate signing, Key encipherment, Digital signature
```

```
Fingerprint:
```

```
8b:84:60:2a:58:5b:80:f0:b9:ae:25:9f:67:3d:d6:81:ee:43:6c:d4 (sha1)
```

```
ab:ec:4d:fe:d4:04:9c:c9:79:1d:9a:33:4e:6d:78:f6 (md5)
```

```
9d:f0:c0:a0:93:74:11:53:d3:4d:2d:75:d3:60:37:5f:fb:b7:a9:67:42:cd:7c:3c:0e:0f:9b:58:36:3c:14:f5  
(sha256)
```

Configuration Statements and Operational Commands

IN THIS CHAPTER

- [Junos CLI Reference Overview](#) | 103

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Learn about the syntax and options that make up the statements and commands and understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- [Junos CLI Reference](#)

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- [Configuration Statements](#)
- [Operational Commands](#)