

Public Key Infrastructure User Guide

Published
2026-02-04

RELEASE

Table of Contents

What is PKI

[PKI Overview](#) | 3

[PKI in Junos OS](#) | 7

[PKI Components in Junos OS](#) | 10

Post-Quantum Cryptography

[PQC Overview](#) | 16

[Get Started with PQC](#) | 18

[The Quantum Buffer](#) | 22

[PQC Signatures for Software Images](#) | 26

Digital Certificates

[Generate Digital Certificates Manually: Configuration Overview](#) | 32

[Digital Certificates in Junos OS](#) | 33

[Generate a Root CA Certificate](#) | 34

[Manually Generate Self-Signed SSL Certificates](#) | 36

[Export Certificates to a Specified Location](#) | 37

[Configure a Root CA Certificate](#) | 37

[Certificate Chain Implementation](#) | 38

[Requirements](#) | 39

[Overview](#) | 39

[SSL Certificate: Configuration Overview](#) | 41

[Configuring Digital Certificates](#) | 44

[Digital Certificates Overview](#) | 44

[Obtaining a Certificate from a CA for an ES PIC](#) | 45

Request a CA Digital Certificate | 45

Generate a Private and Public Keypair for Digital Certificates for an ES PIC | 46

Requesting a CA Digital Certificate | 46

Request a CA Digital Certificate | 46

Generate a Public-Private Keypair | 47

Generate and Enroll a Local Digital Certificate | 47

Apply Local Digital Certificate to an IPsec Configuration | 47

Configure Automatic Reenrollment of Digital Certificates | 48

Configure Digital Certificates for an ES PIC | 48

Configure the CA Properties for an ES PIC | 50

Specify the CA Name | 51

Configure the CRL | 51

Configure the Type of Encoding Your CA Supports | 51

Specify an Enrollment URL | 52

Specify a File to Read the Digital Certificate | 52

Specify an LDAP URL | 52

Configure the Cache Size | 53

Configure the Negative Cache | 53

Configure the Number of Enrollment Retries | 54

Configure the Maximum Number of Peer Certificates | 54

Configure the Path Length for the Certificate Hierarchy | 54

Certificate Authority

Certificate Authority Profiles | 55

Configure CA Profiles | 56

Configure a Trusted CA Group | 58

Create a Trusted CA Group | 59

Delete a CA Profile from a Trusted CA Group | 60

Delete a Trusted CA Group | 61

Example: Configure a CA Profile | 62

Requirements | 62

Overview | 62

Configuration | 62

Verification | 64

Example: Configure an IPv6 Address as the Source Address for a CA Profile | 64**Self-Signed Digital Certificates****Self-Signed Certificates | 66****Example: Generate a Public-Private Keypair | 67**

Requirements | 67

Overview | 67

Configuration | 67

Verification | 68

Manually Generate Self-Signed SSL Certificates | 68**Example: Manually Generate Self-Signed Certificates | 69**

Requirements | 69

Overview | 69

Configuration | 69

Verification | 70

Manage Automatically Generated Self-Signed Certificates | 70**Enable HTTPS and XNM-SSL Services on Switches Using Self-Signed Certificates (CLI Procedure) | 71****Enroll a Certificate****Enroll Digital Certificates Online: Configuration Overview | 73****Certificate Enrollment | 73**

Online CA Certificate Enrollment | 74

Local Certificate Requests | 74

CMPv2 and SCEP Certificate Enrollment | 74

Example: Manually Generate a CSR for the Local Certificate | 75

 Requirements | 75

 Overview | 75

 Configuration | 76

 Verification | 76

Example: Manually Load CA and Local Certificates | 77

 Requirements | 77

 Overview | 77

 Configuration | 78

 Verification | 78

Configure PKI and SSL Forward Proxy to Authenticate Users | 79

Configure Digital Certificates for Adaptive Services Interfaces | 80

 Configure the Certificate Authority Properties | 81

 Configure the Certificate Revocation List | 83

 Manage Digital Certificates | 84

 Configure Auto-Reenrollment of a Router Certificate | 87

Enroll a CA Certificate Using SCEP | 90

Enroll a CA Certificate Online Using SCEP | 90

Example: Enroll a Local Certificate Online Using SCEP | 91

 Requirements | 91

 Overview | 92

 Configuration | 92

 Verification | 93

Example: Using SCEP to Automatically Renew a Local Certificate | 93

 Requirements | 94

 Overview | 94

 Configuration | 95

 Verification | 95

Enroll a CA Certificate Online Using CMPv2 | 96

Install a Digital Certificate on Your Router | 98

Manually Request a Digital Certificate | 98

Understand ACME Protocol | 102

What is ACME Protocol | 102

Enroll Local Certificate Using Let's Encrypt Server | 103

Manually Reenroll Local Certificate | 105

Delete ACME Account | 105

Platform-Specific SSL Termination Services Behavior | 105

Revoke a Certificate

Example: Manually Load a CRL onto the Device | 106

Requirements | 107

Overview | 107

Configuration | 107

Verification | 108

Dynamic CRL Download and Verify | 108

Example: Configuring a CA Profile with CRL Locations | 111

Requirements | 111

Overview | 112

Configuration | 112

Verification | 112

Example: Verify Certificate Validity | 113

Requirements | 113

Overview | 113

Configuration | 113

Verification | 114

Delete a Loaded CRL | 114

Validate a Certificate

[Validate Digital Certificate on SRX Series Firewall | 115](#)

[Validate Digital Certificate on MX Series Devices | 121](#)

[Example: Validating Digital Certificate by Configuring Policy OIDs | 127](#)

[Requirements | 128](#)

[Overview | 128](#)

[Configuration | 128](#)

[Verification | 130](#)

[Example: Improving Digital Certificate Validation by Configuring Policy OIDs on an MX Series Device | 132](#)

[Requirements | 133](#)

[Overview | 133](#)

[Configuration | 133](#)

[Verification | 135](#)

[Example: Configuring a Device for Peer Certificate Chain Validation | 137](#)

[Requirements | 137](#)

[Overview | 138](#)

[Configuration | 139](#)

[Verification | 146](#)

[IKE and IPsec SA Failure for a Revoked Certificate | 148](#)

[Configure the Certificate Expiration Trap | 149](#)

Update a Certificate

[Dynamic Update of Trusted CA Certificates | 151](#)

[Configure Dynamic Update of Trusted CA Certificates | 152](#)

[Check Connectivity to the CDN Server | 153](#)

[Enable Automatic Download of Default Trusted CA Certificates | 154](#)

[Download Default Trusted CA Certificates Automatically | 155](#)

Download Default Trusted CA Certificates Manually | 156

Check the Download Status of Default Trusted CA Certificates | 157

Deactivate Automatic Download of Trusted CA Certificates | 158

Delete a Certificate

Delete a Loaded CRL | 159

Configuration Statements and Operational Commands

Junos CLI Reference Overview | 160

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.

Table 1: Configure PKI in Junos OS

Task	Description
Create CA profile	<p>Define CA profile attributes: Create a CA profile to specify the CA settings, including the CA identity and any additional attributes required.</p> <p>Specify enrollment parameters: Configure the enrollment retry value and the time interval between attempts to automatically enroll the CA certificates online.</p> <p>Set revocation check: Specify the certificate revocation list (CRL) refresh interval and URL for revocation checks.</p> <p>See Certificate Authority.</p>
Generate certificate	<p>Generate certificate request: Generate a public or private keypair and then create the certificate request using the keypair.</p> <p>Send certificate request: Send the certificate request to the CA administrator through an email or an out-of-band method. Specify an email address for the CA administrator if needed.</p> <p>See Self-Signed Digital Certificates.</p>

Table 1: Configure PKI in Junos OS (Continued)

Task	Description
Load CA and local certificates	<p>Load CA certificate: Load the CA certificate from an external file and associate it with the configured CA profile.</p> <p>Load local certificate: Load the local certificate into local storage from the specified external file, ensuring proper linkage with the private or public keypair.</p> <p>See Enroll a Certificate.</p>
Configure IPsec VPN with certificates	<p>Define IKE policy and gateway: Configure the IKE policy and gateway to use RSA-signature authentication method and the local and CA certificates.</p> <p>See Configure Multiple Certificate Types to Establish IKE and IPsec SA.</p>

What is PKI

IN THIS SECTION

- [PKI Overview | 3](#)
- [PKI in Junos OS | 7](#)
- [PKI Components in Junos OS | 10](#)

PKI Overview

SUMMARY

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

IN THIS SECTION

- [Introduction to PKI | 3](#)
- [How PKI Works | 3](#)
- [Benefits of PKI | 5](#)
- [PKI Terminology | 6](#)

Introduction to PKI

PKI provides a way of verifying the identity of a remote site by using a digital certificate. PKI uses a CA to validate and digitally sign your information. This process ensures that neither your information nor the signature can be modified. Once you sign your information, the information becomes a digital certificate. Devices that receive a digital certificate verify the certificate's information by validating the signature with public key cryptography.

The PKI consists of the following components for managing digital certificates:

- RA: Verifies the identities of entities, authorizes their certificate requests, and generates unique asymmetric keypair (unless the users' certificate requests already contain public keys).
- CA: Issues corresponding digital certificates for the requesting entities.
- CRL: Identifies the certificates that are no longer valid. Each entity possessing the authentic public key of a CA can verify the issued certificates.

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](#)

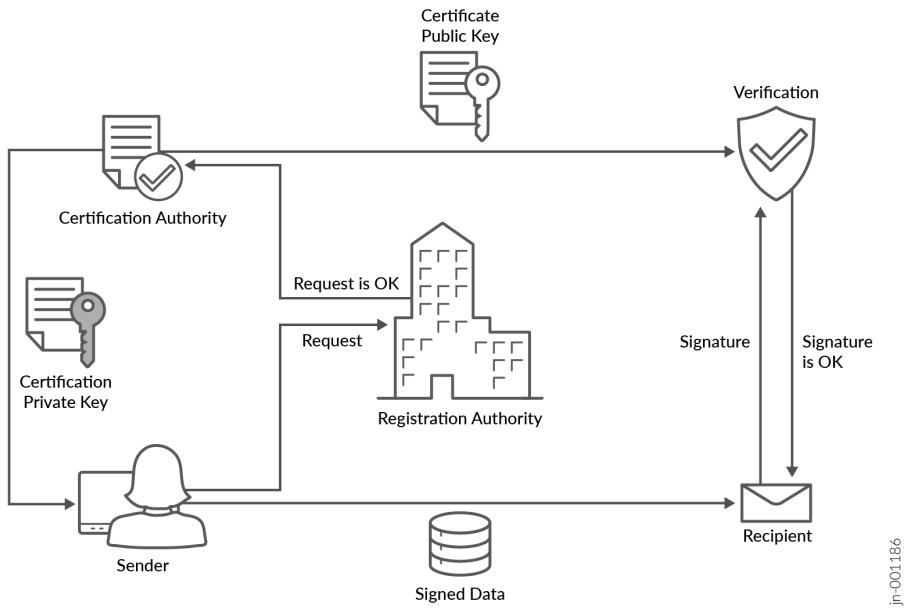


Table 2: PKI Components

PKI Key Components	Description
CA	A trusted third-party organization that creates, enrolls, validates, and revokes digital certificates. The CA guarantees a user's identity, issues public and private keys for message encryption and decryption.
Registration authority (RA)	Verifies the identities of entities, authorizes their certificate requests, and generates unique asymmetric keypair unless the users' certificate requests already contain public keys.
Digital certificates	Electronic documents that contain information about the entity, such as a VPN gateway. The CA signs the digital certificates to ensure their authenticity and integrity.
Public and private keys	A pair of keys used in public key cryptography, generated simultaneously, and linked mathematically. The public key is used for encryption, while the private key is used for decryption. These keys are

Table 2: PKI Components (Continued)

PKI Key Components	Description
IKE and PKI	During IKE Phase 1 setup, a certificate can identify the peer by IP address, FQDN, user FQDN (U-FQDN), or DN. The CA adds the IKE ID to the <i>SubjectAlternativeName</i> field of the certificate.
Certificate LCM	<p>Includes phases such as:</p> <ul style="list-style-type: none"> • Public and private keys generation • Identity information • Enrollment (request and retrieval) • Usage within IKE • Certificate validation and revocation checks • Certificate renewal

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 CAs, 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 authenticity verification 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 Terminology

Table 3: Terminology

Term	Description
PKI	A framework that enables secure, encrypted communications and digital signature services.
CA	An entity that issues digital certificates.
Digital certificate	A digital form of identification issued by a CA that verifies authenticity.
CRL	A list of certificates that have been revoked by a CA before their expiration date.
Enrollment	The process of requesting and receiving a digital certificate from a CA.
Keypair	A pair of cryptographic keys (public and private) used for encryption and decryption.
Root certificate	The top-most certificate in the certificate chain, issued by a root CA.
Self-signed certificate	A certificate that is signed by the system creating it, rather than a trusted CA creating it.
Private key	The secret part of the keypair used in asymmetric encryption.
Public key	The non-secret part of the keypair used in asymmetric encryption.
Digital signature	A mathematical scheme for verifying the authenticity of digital messages or documents.

Table 3: Terminology (Continued)

Term	Description
Certificate chain	A sequence of certificates, where each certificate is signed by the subsequent CA.

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 | 7](#)
- [Basic Elements of PKI in Junos OS | 7](#)
- [Certificates | 9](#)

PKI Applications Overview

Junos OS uses public and private keys in the following areas:

- SSH and SCP for secure CLI-based administration
- SSL for secure Web-based administration and for https-based webauth for user authentication
- IKE for IPsec VPN tunnels.

Note the following points:

- Currently, Junos OS supports only IKE using PKI certificates for public key validation.
- The SSH and SCP are used exclusively for system administration. Junos OS uses the OOB fingerprints for public key identity binding and validation.

Basic Elements of PKI in Junos OS

Junos OS supports three specific types of PKI objects.

Table 4: Elements of PKI in Junos OS

Elements of PKI	Description
Private and public keypair	<p>A private and public keypair is a fundamental component used for secure communication.</p> <ul style="list-style-type: none"> • Public Key: The public key is used to encrypt data. It is published and can be shared with others without compromising security. Data encrypted with the public key can only be decrypted using the corresponding private key. • Private Key: The private key is used to decrypt data that was encrypted with the public key. It is kept secret and should not be shared with anyone. <p>By using the private and public keypair, Juniper Networks devices can establish secure connections and protect data transferred over public networks.</p>
Certificates	<ul style="list-style-type: none"> • 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 device. • Pending certificate—A pending certificate contains a keypair and identity information that is generated into a PKCS10 certificate request and manually sent to a CA. While the Juniper Networks device waits for the certificate from the CA, the existing object (keypair and the certificate request) is tagged as a certificate request or pending certificate. • CA certificates—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.

Table 4: Elements of PKI in Junos OS (*Continued*)

Elements of PKI	Description
CRL	A CRL in PKI is a time-stamped list of digital certificates that have been revoked by a CA. This list is signed by the CA and made available to participating peers on a regular periodic basis. In Junos OS, you can configure CRLs to ensure that certificates are not used if certificates have been revoked.

Certificates

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 this 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. The PKCS12 files 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 checks enable and 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 | 10](#)
- [IKE | 10](#)
- [PKI over HTTPS | 11](#)
- [Trusted CA Group | 11](#)
- [Cryptographic Key Handling | 11](#)
- [Certificate Signatures and Verification | 12](#)
- [Certificate Validation | 13](#)

PKI Management and Implementation

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

- CA certificates and authority configuration
- Local certificates including the device's identity (for example: IKE ID type and value)
- Private and public keys
- CRL for certification validation

Junos OS supports three different types of PKI objects:

IKE

The procedure for digitally signing messages sent between two participants in an 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 keypair, the participants use the sender's public-private keypair.

PKI over HTTPS

HTTPS support for PKI enhances the security of certificate management operations. The HTTPS on the PKI establishes secure communication channels for SCEP enrollment and CRL revocation, protecting sensitive information.

The PKI process dynamically selects HTTP or HTTPS based on configured URLs, providing flexibility and secure transmissions.

- If you use a HTTP URL in CA profile, PKID process uses the HTTP communication channel.
- If you use a HTTPS URL in CA profile, PKID process uses the HTTPS communication channel.

Trusted CA Group

A CA is a trusted third party responsible for issuing and revoking certificates. A trusted CA group is a group of multiple CAs (CA profiles) in one 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, let's assume you have two endpoints, endpoint A and endpoint B, 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 respective endpoint gets validated. If not, the authorization rejects the connection.

Cryptographic Key Handling

Cryptographic key handling stores persistent keys in device memory without altering the keys. The internal memory is not accessible to adversaries. You can enable key handling mechanism for cryptographic keys to provide security.

When you enable cryptographic key handling:

- The cryptographic key handling encrypts keys when not immediately in use.
- Performs error detection when copying a key from one memory location to another.

- Overwrites the memory location of a key with a random bit pattern when the key is no longer in use.

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

Certificate Signatures and Verification

A digital certificate is an electronic means for verifying your identity through a trusted third party, known as a 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 that CA for the addresses of their CA and CRL servers (for obtaining certificates and CRLs) and for the information they require when submitting personal certificate requests. When you're 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.

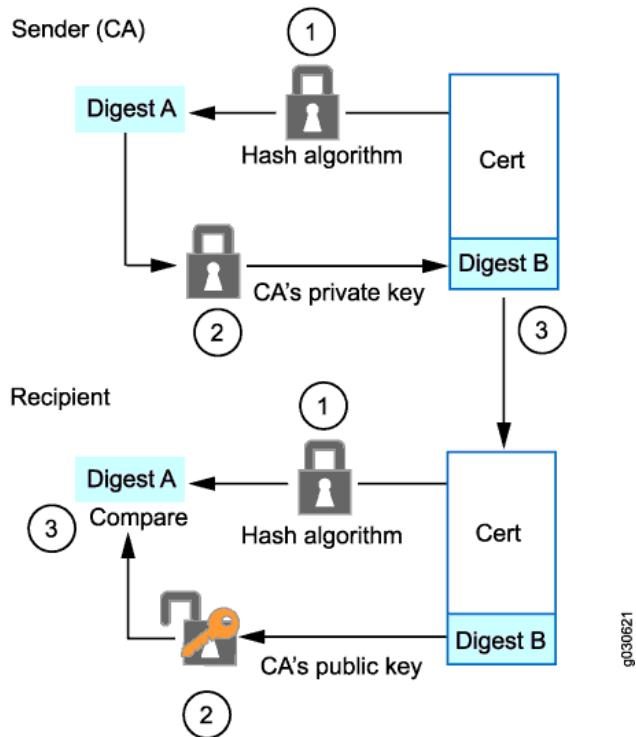
The recipient of the certificate generates another digest by applying the same hash algorithm to the certificate file, then uses the CA's public key to decrypt the digital signature. By comparing the decrypted digest with the newly generated digest, the recipient can confirm the integrity of the CA's signature and, by extension, the integrity of the accompanying certificate. [Figure 2 on page 13](#) illustrates this entire 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 CRL.

Figure 2: Digital Signature Verification

1. Using a hash algorithm, the CA generates digest A from the certificate.
2. Using the private key, the CA encrypts digest A. The result is digest B, the digital signature.
3. The CA sends the digitally signed certificate to the person who requested it.

1. Using a hash algorithm, the recipient generates digest A from the certificate.
2. Using the CA's public key, the recipient decrypts digest B.
3. The recipient compares digest A with digest B. If they match, the recipient knows that the certificate has not been tampered with.



When 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 ECDSA signatures are used, SHA-256 is used for ECDSA-256 signatures and SHA-384 is used for ECDSA-384 signatures.

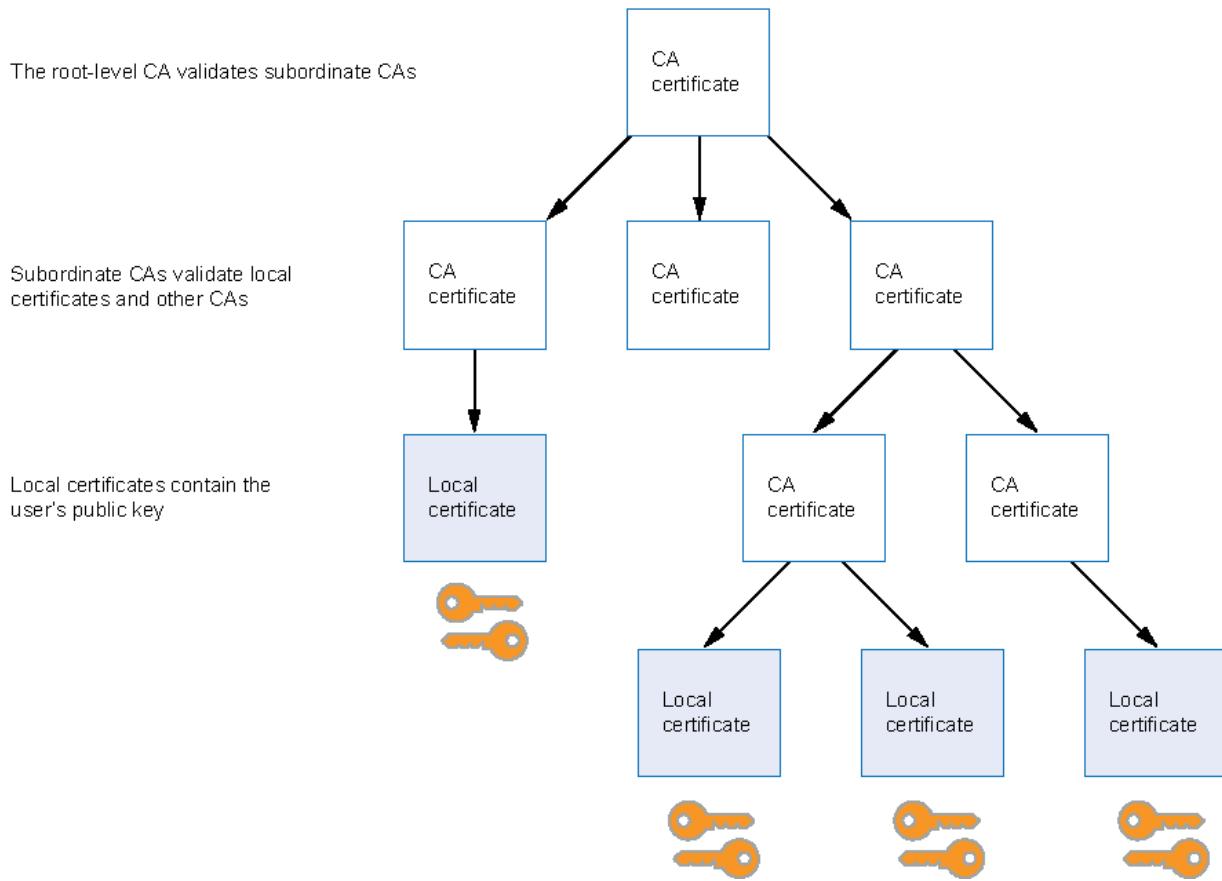
Starting in Junos OS Release 18.1R3, the default hash algorithm that is used for validating automatically and manually generated self-signed PKI certificates is SHA-256. Before Junos OS Release 18.1R3, SHA-1 is used as default hash algorithm.

Certificate Validation

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

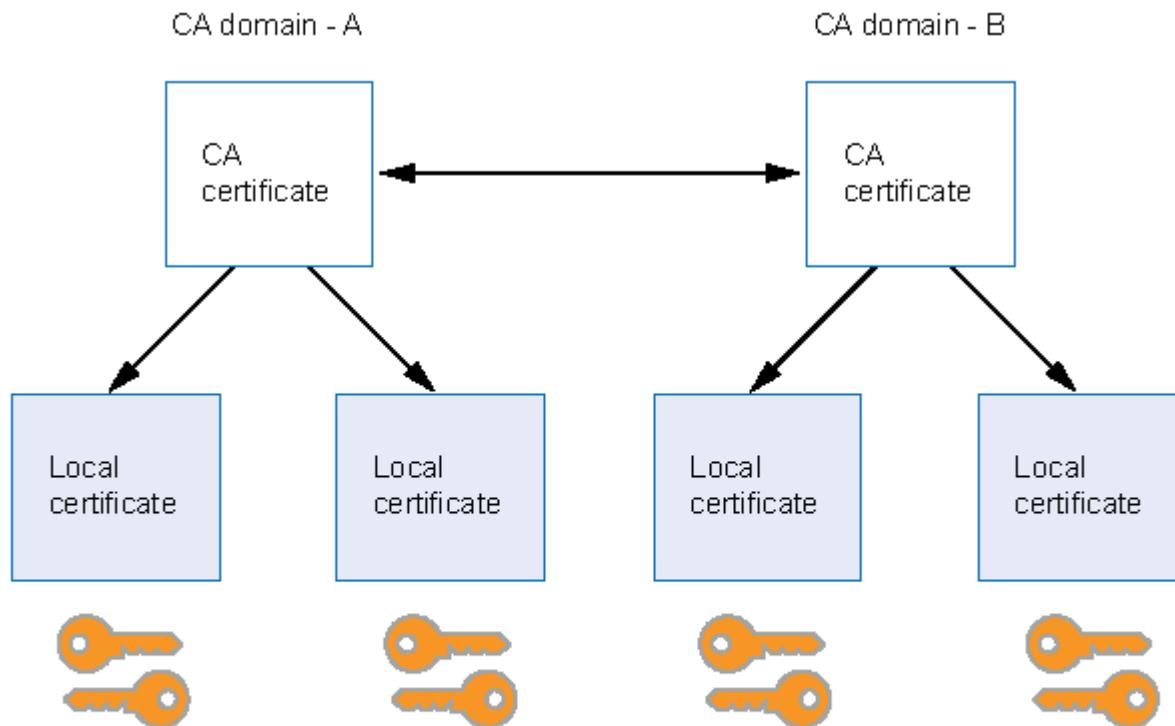
Figure 3 on page 14 shows the structure of a single-domain CA with multiple hierarchy levels.

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 15](#).

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.

Post-Quantum Cryptography

IN THIS SECTION

- [PQC Overview | 16](#)
- [Get Started with PQC | 18](#)
- [The Quantum Buffer | 22](#)
- [PQC Signatures for Software Images | 26](#)

PQC Overview

SUMMARY

Learn about the quantum threat and why post-quantum cryptography (PQC) matters for security.

IN THIS SECTION

- [Quantum Computing Threat | 16](#)
- [Why PQC Matters | 17](#)
- [Benefits | 18](#)

Quantum Computing Threat

The quantum computing threat refers to the potential security risk that powerful quantum computers pose to traditional cryptographic systems, which primarily depend on classical computing methods.

Classical computers perform complex mathematical operations using binary operations—zeros and ones. For classical computers, calculating discrete logarithms for very large prime numbers is computationally impossible. As a result, security experts use the Diffie-Hellman (DH) algorithm or Elliptic Curve Diffie Hellman (ECDH) algorithm as a trusted method for secure key exchange. Similarly, RSA and Elliptic Curve Digital Signature Algorithm (ECDSA) algorithms are trusted methods for creating digital signatures.

Quantum computers work differently using qubits. A qubit can be 0, 1, or both at the same time due to superposition. This capability allows quantum computers to explore many possibilities at once, providing more processing power for solving certain problems. Quantum computers solve certain tasks much faster than classical computers, but they introduce new risks for traditional cryptography.

Traditional cryptographic algorithms such as RSA, Diffie-Hellman (DH), and Elliptic Curve Cryptography (ECC) depend on mathematical problems that classical computers can't solve efficiently. Quantum computers can solve those problems, including discrete logarithms and factorization, much faster. Algorithms such as Shor's and Grover's enable quantum computers to compromise classical cryptography. Shor's algorithm can break DH much faster than any traditional digital computer. Grover's algorithm can reduce the security of symmetric encryption. The National Institute of Standards and Technology (NIST) considers larger key sizes, such as AES-256, secure.

Cryptographically Relevant Quantum Computers (CRQCs) are powerful quantum computers capable of breaking traditional public key cryptography. CRQCs do not exist today, but could be developed in the future.

Organizations should prepare for this shift by adopting post-quantum cryptography (PQC) standards and implementing quantum-resistant strategies to protect sensitive data from attacks by quantum computers.

Why PQC Matters

Quantum resistance is the ability of a cryptographic algorithm to remain secure even against an adversary computing method. Quantum-resistant cryptographic algorithms can withstand attacks from quantum computers. These algorithms use mathematical problems that are hard to solve even for powerful quantum computers like CRQCs.

PQC refers to these quantum-resistant algorithms that protect against quantum computing threats. PQC is an evolutionary enhancement of today's public key infrastructure (PKI). PQC integrates into PKI to ensure it remains secure and trustworthy in the post-quantum era. PQC maintains long-term confidentiality, integrity, and authentication needs for secure communication.

The NIST addresses the quantum threat by standardizing PQC algorithms for global compliance. [Table 5 on page 17](#) shows NIST-approved PQC standards as Federal Information Processing Standards (FIPS) specifications.

Table 5: NIST PQC Specifications

Specification	Algorithm	Use Case
FIPS 203	Module-Lattice-Based Key Encapsulation Mechanism (ML-KEM), based on CRYSTALS-Kyber	Encryption for secure web applications.
FIPS 204	Module-Lattice-Based Digital Signature (ML-DSA), based on CRYSTALS-Dilithium	Authentication using digital signatures and detection of unauthorized access.
FIPS 205	Stateless Hash-Based Digital Signature (SLH-DSA), based on SPHINCS+	Robust authentication using digital signatures for long-term security.

The Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) promotes the adoption of quantum-resistant PQC algorithms using the strongest standardized key sizes—ML-DSA-87 and ML-KEM-1024. These algorithms align with NIST PQC standards to protect encryption, key exchange, and digital signatures against quantum attacks. CNSA 2.0 also recommends combining classical and PQC algorithms to enable gradual migration without disrupting operations.

In addition to PQC, Juniper Networks supports quantum-safe security technologies such as Quantum Key Distribution (QKD). Juniper integrates QKD with quantum-safe IPsec and MACsec implementations, offering an alternative approach to securing data against quantum computing threats.

Benefits

- Quantum safe security—Mitigates vulnerabilities in classical cryptography and ensures resilience against attacks by quantum computers.
- Protection against future threats—Defends against 'harvest now, decrypt later' threat model where attackers store encrypted data today for future decryption using quantum computers.
- Modernize PKI—Complements PKI by integrating the quantum-resistant algorithms into the existing infrastructure.

RELATED DOCUMENTATION

[PKI Overview](#)

[Get Started with PQC](#)

Get Started with PQC

SUMMARY

Read this topic to learn about PQC features in Junos OS and Junos OS Evolved.

IN THIS SECTION

- [Quantum-Resistant Security Features | 18](#)
- [Assess Your Quantum-Resistant Security Requirements | 21](#)
- [Next Steps | 21](#)

Quantum-Resistant Security Features

Juniper Networks provides advanced cryptographic capabilities to safeguard networks against emerging quantum threats. Junos® OS and Junos® OS Evolved use the following approaches to ensure protection:

- Implement quantum-safe technologies such as the ETSI Quantum Key Distribution (QKD) to interface with key delivery servers for secure key exchange in VPNs.
- Adapt cryptoagility with Quantum Buffer to meet the evolving cryptographic standards and enable flexible key management through fungible key delivery servers.

- Adopt the National Institute of Standards and Technology (NIST)-approved post-quantum cryptography (PQC) algorithms for encryption, authentication, and key exchange.

Juniper Networks PQC implementation provides seamless integration with backward compatibility and no service disruption.

Use [Feature Explorer](#) to confirm platform and release support for specific features.

Learn about the following key features and implementations.

Quantum-Safe Support

[Table 6 on page 19](#) shows a list of features supported with quantum-safe implementations in Junos OS and Junos OS Evolved.

Table 6: Quantum-Safe Support

Feature Name	Details
Quantum-Safe IPsec VPN	<p>Supports RFC 8784 in IPsec VPNs.</p> <ul style="list-style-type: none"> • Uses out-of-band key exchange mechanisms, such as QKD, to strengthen IPsec tunnel security. • Implements preshared key mixing, also known as post-quantum preshared keys (PPKs), in Internet Key Exchange Version 2 (IKEv2) for post-quantum security. <p>See Quantum Safe IPsec VPN.</p>
Quantum-Safe MACsec	<ul style="list-style-type: none"> • Implements frequent key refresh through QKD using key delivery servers. • Uses AES-256 encryption, which is inherently quantum-safe.

PQC Libraries and Algorithms

[Table 7 on page 20](#) shows NIST-standardized quantum-resistant PQC algorithms.

Table 7: NIST-Standardized PQC algorithms

Algorithm	Type	Description
ML-KEM	Key Encapsulation	Module-lattice-based key encapsulation mechanism built on CRYSTALS-Kyber algorithm.
ML-DSA	Digital Signature	Module-lattice-based digital signature based on CRYSTALS-Dilithium algorithm.
SLH-DSA	Digital Signature	Stateless hash-based digital signature based on SPHINCS+ algorithm.

Juniper Networks includes PQC libraries to ensure readiness for PQC algorithms.

[Table 8 on page 20](#) shows the list of supported cryptographic libraries for PQC.

Table 8: Supported PQC Libraries

Library	Purpose
OpenSSL 3.5	Provides support for PQC algorithms for secure communications.
Leancrypto	Serves as the kernel cryptographic library for PQC readiness.

Quantum Buffer

Juniper Networks strengthens finite field cryptography (FFC) parameters in existing public key cryptography. This approach reduces the risk from Cryptographically Relevant Quantum Computers (CRQCs) and supports a smooth transition to PQC. This Quantum Buffer approach extends the time window and resists cryptanalytic attacks by reinforcing FFC parameters.

Juniper Networks implements Quantum Buffer for the SSH protocol. See "[The Quantum Buffer](#)" on page 22.

Quantum-Resistant Key Exchange in SSH

Juniper Networks supports Shor-resistant key exchange to strengthen SSH security against quantum attacks. Network devices can use the hybrid Streamlined NTRU Prime 761 plus X25519 key exchange

algorithm. This algorithm combines lattice-based and elliptic curve techniques to provide strong protection against quantum threats. See [key-exchange](#).

PQC-Signed Software Images

Juniper Networks provides PQC-signed Junos OS and Junos OS Evolved images. These images use digital signatures for integrity verification that comply with the Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) recommendations. You can validate the image signature off-box using your preferred verifier on a local machine. See ["PQC Signatures for Software Images" on page 26](#).

Assess Your Quantum-Resistant Security Requirements

Organizations face evolving security challenges as quantum computing advances. To protect your network and maintain compliance, understand your security requirements and implement appropriate quantum-resistant security features. Use the following assessment to determine which quantum-resistant security features are essential for your network:

- **Asses infrastructure**—Assess your network infrastructure including your Juniper Networks hardware and software. Use PQC-signed software images as first line of defense to ensure image integrity and compliance with CNSA 2.0 recommendations. See ["PQC-Signed Software Images" on page 21](#).
- **Explore PQC readiness and quantum-safe options**—Understand Juniper Networks readiness with PQC algorithms. Review the available technologies for quantum-safe VPNs, including quantum-safe IPsec and quantum-safe MACsec. See ["PQC Libraries and Algorithms" on page 19](#), ["Quantum-Resistant Key Exchange in SSH" on page 20](#), and ["Quantum-Safe Support" on page 19](#).
- **Evaluate risk tolerance**—Know your network's risk tolerance for quantum computing threats. Implement cryptoagility with Quantum Buffer solution and fungible key management systems to strengthen existing cryptography during migration. See ["Quantum Buffer" on page 20](#) and ["Quantum-Safe Support" on page 19](#).
- **Secure critical deployments**—Determine your critical deployments such as IPsec and MACsec implementations. Secure these deployments with quantum-safe technologies. See ["Quantum-Safe Support" on page 19](#).

Plan your upgrade for a quantum-resistant network.

Next Steps

Read the following topics to know more about the quantum-resistant security features:

- ["The Quantum Buffer" on page 22](#)
- ["PQC Signatures for Software Images" on page 26](#)

- [Quantum Safe IPsec VPN](#)

RELATED DOCUMENTATION

[PQC Overview](#)

The Quantum Buffer

SUMMARY

Read this topic to learn about the Juniper Networks Quantum Buffer feature to protect network devices from quantum threats and keep the network secure for longer periods.

IN THIS SECTION

- [Overview | 22](#)
- [Benefits | 23](#)
- [Limitations | 23](#)
- [Quantum Buffer for SSH | 23](#)
- [Configure Quantum Buffer for SSH | 24](#)

Overview

Cryptographically relevant quantum computers (CRQCs), if realized, could significantly threaten traditional public key cryptography. In response, the industry is transitioning toward post-quantum cryptographic algorithms that can be resistant to quantum attacks. Although NIST has standardized these algorithms, they require further time to match the maturity and reliability of traditional cryptographic methods.

To mitigate the potential risk posed by CRQCs and facilitate a smoother transition to post-quantum cryptography (PQC), Juniper Networks provides a solution to enhance existing public key cryptographic systems. The solution involves strengthening the finite field cryptography (FFC) parameters, thereby extending the time window during which current systems remain secure against quantum attacks. The Quantum Buffer feature provides the capability to resist cryptanalytic attacks against traditional public key cryptographic algorithms by improving the strength of their parameters.

Juniper Networks implements Quantum Buffer for the SSH protocol.

Benefits

- Improved resiliency—Enhances system resilience against evolving quantum attacks and provides a phased approach to adopting PQC, thereby mitigating operational risks associated with the transition.
- Enhanced defense—Provides enhanced protection to the classical public key cryptography in the event of a CRQC. The feature also protects against advanced attacks on classical cryptography, including attacks targeting discrete log vulnerabilities.
- Cryptoagility—Provides cryptographic agility without disrupting your existing infrastructure.
- Robust solution—Facilitates continuous strengthening of security with the ability to periodically update Diffie-Hellman (DH) moduli.
- Optimized performance—Balances enhanced security with performance, minimizing resource overhead while meeting the protocol compliance requirements.

Limitations

- Quantum Buffer provides a time-based buffer for a CRQC scenario offering time-limited security if a CRQC emerges in the future.
- The prime-generation algorithms consume significant computing resources and are nondeterministic in nature. Therefore, it is difficult to accurately predict the time and CPU usage required for generating each prime.

Quantum Buffer for SSH

Juniper Networks devices establish secure SSH sessions for remote management, configuration, and automation. Juniper Networks implements JSSH with the Quantum Buffer for SSH.

The Quantum Buffer for SSH supports:

- Dynamic generation of Finite Field Diffie-Hellman (FFDH) moduli that have large prime subgroups.
- Periodic update of FFDH prime moduli based on the configurable refresh interval.
- Ability to optionally specify the number of prime moduli the device must generate for added protection.

The qbufd process manages the generation of prime moduli. JSSH uses these moduli for the FFDH group exchange algorithms, group-exchange-sha1 and group-exchange-sha2.

JSSH uses the prime moduli that the qbufd process generates for the Diffie-Hellman key exchange.

Use the following recommendations when using Quantum Buffer:

- As the qbufd process might cause the device to run at high CPU utilization, specify a larger refresh interval in the configuration to mitigate the issue.
- Use large prime group and subgroup sizes in Diffie-Hellman designer-type moduli. We recommend a minimum size of 2048 bits and an optimal size of 3072 or 4096 bits for a subgroup. Specify the count as 3 or 4 for better protection.

If you don't configure the moduli, JSSH uses the default fixed moduli set of safe primes.

Configure Quantum Buffer for SSH

Prerequisites

To configure Quantum Buffer for SSH, complete the following prerequisites:

- See [Feature Explorer](#) to confirm platform and release support for specific features.
- Set up your Juniper Networks device with basic configuration including interface setup and user creation.

Configuration

To configure Quantum Buffer for SSH:

1. Specify the cryptographic moduli settings required for the key exchange algorithms.

```
[edit]
user@host# set system services ssh moduli type designer refresh monthly count 3
```

In this command, the SSH service uses designer-type moduli by regenerating three new moduli each month with every refresh. For more details, see [moduli](#).

2. Configure the SSH group exchange algorithm. Note that the default algorithm set for JSSH includes the DH group exchange algorithm group-exchange-sha2.

To use group-exchange-sha1, configure the algorithm:

```
[edit]
user@host# set system services ssh key-exchange group-exchange-sha1
```

To use both group-exchange-sha1 and group-exchange-sha2, configure both the algorithms:

```
[edit]
user@host# set system services ssh key-exchange [group-exchange-sha1 group-exchange-sha2]
```

Ensure your SSH client requests one of these algorithms when connecting to the device.

Verification

To verify Quantum Buffer for SSH:

1. Verify that the device runs the qbufd process.

You can verify that the process is running by configuring the syslog to include qbufd events.

```
[edit]
user@host# set system syslog file qbufd.log any any
user@host# set system syslog file qbufd.log match qbufd
```

In this command, *qbufd.log* is the log file name, but you can specify any name.

Alternatively, you can also verify the process at the device's shell prompt.

```
[edit]
user@host:~# ps aux | grep qbufd
```

2. Verify moduli generation.

```
user@host:~# grep "Moduli generated" /var/log/qbufd.log
```

As moduli generation takes time, monitor the logs and the qbufd process. When the qbufd process stops running, you'll find the new prime moduli details at /etc/ssh/moduli on your device.

3. Verify that the SSH login with the configured key exchange algorithm is successful.

Run the following command in your SSH client such as a laptop.

```
localhost:~$ ssh -o kexAlgorithms=diffie-hellman-group-exchange-sha256 user@host
```

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
25.4R1-EVO	Support introduced for Quantum Buffer for SSH in Junos OS Evolved Release 25.4R1.

RELATED DOCUMENTATION

[moduli](#)

[key-exchange](#)

[Remote Access Overview \(Junos OS Evolved\)](#)

PQC Signatures for Software Images

SUMMARY

Read this topic to learn about PQC-signed Junos OS and Junos OS Evolved software images and how to verify the PQC signatures.

IN THIS SECTION

- [Overview | 26](#)
- [Benefits | 27](#)
- [Limitations | 27](#)
- [About PQC Signature Verification \(Off-Box\) | 27](#)
- [Verify PQC Signatures \(Off-Box\) | 29](#)

Junos® operating system (Junos OS) and Junos® OS Evolved are operating systems (OS) for Juniper Networks devices. Junos OS runs on FreeBSD, while Junos OS Evolved runs natively on Linux designed for cloud-scale environments. As the first line of defense against quantum threats, Juniper ensures the authenticity of these software images using post-quantum cryptography (PQC) signatures.

See [Feature Explorer](#) to confirm platform and release support for specific features.

Overview

Quantum-resilient software image signing uses PQC to protect Junos OS and Junos OS Evolved software images against threats from quantum computing. The Commercial National Security Algorithm

Suite 2.0 (CNSA 2.0) recommends using a PQC algorithm, such as ML-DSA-87, for protection against quantum attacks. In line with the guidance, Juniper Networks signs software images by using:

- ML-DSA-87 PQC algorithm for digital signatures
- SHA-512 for hashing

While both SHA-384 and SHA-512 are permitted under CNSA 2.0, Juniper Networks adopts SHA-512 for stronger cryptographic protection according to the guidance.

Juniper Networks image signing infrastructure is robust and supports secure PQC signing. You can use the following method to verify the PQC signature when you download the image:

- Off-box PQC signature verification

This implementation aligns with NIST PQC standards and follows CNSA 2.0 guidance, ensuring compliance with evolving security standards.

Benefits

- Robust key management—Juniper Networks image signing infrastructure provides robust and secure key management with reinforced public/private keypair for distinct deployment environments.
- Image integrity adherence—Juniper Networks signs image hashes using the recommended PQC algorithms to prevent tampering, ensure authenticity, and protect against quantum computing threats.
- Streamlined signature requests—You can request PQC signatures and public keys through customer support for faster and more efficient image verification.
- Simplified verification process—Using tools of your choice such as OpenSSL enables independent validation for checking the integrity of downloaded images.

Limitations

- Protection against attacks by cryptographically relevant quantum computers (CRQC) on the device isn't in scope.
- No support for on-box image validation.

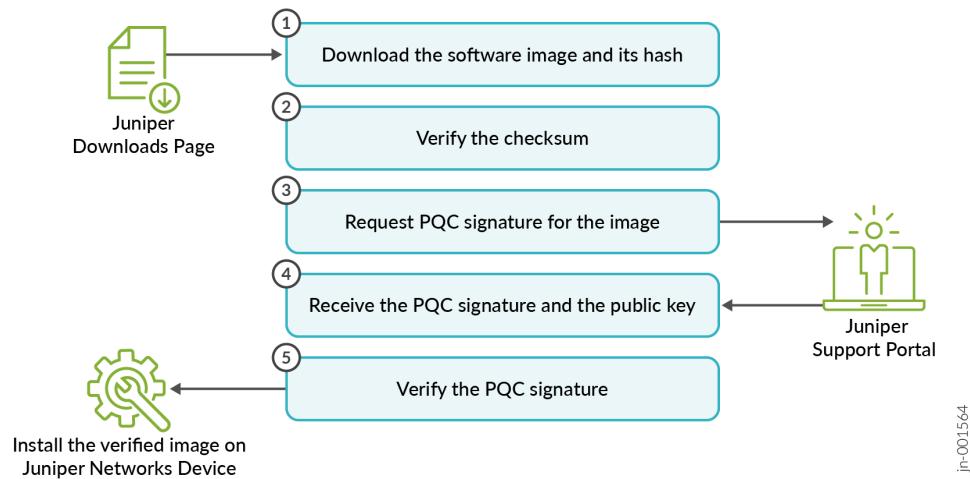
About PQC Signature Verification (Off-Box)

In the off-box PQC signature verification process, you perform the PQC digital signature verification for Junos OS or Junos OS Evolved software images on a local computer. PQC signatures provide additional security beyond the existing legacy signatures. PQC signatures don't replace legacy signatures.

Regardless of the verification success, you can decide whether to use the image. You can verify the image authenticity using PQC signature only if you've downloaded a PQC-signed image.

To use this verification method, check the [Technical Support Bulletin](#) for the availability of a PQC signature.

Figure 5: PQC Signature Validation Workflow (Off-Box Method)



As shown in [Figure 5 on page 28](#), the PQC signature verification workflow includes the following steps:

1. Download the software image and its hash from the [Juniper Downloads Page](#) to your local machine.
2. On your local machine, verify the SHA-512 checksum of the downloaded software image before verifying the PQC signature.
3. Send a request from the [Juniper Support Portal](#) to fetch the PQC signature of the software image.
4. You'll receive a link to download the PQC signature and the associated public key from the Juniper Support Portal.
5. Use a verifier to confirm the PQC signature to validate the software image. See ["Verify PQC Signatures \(Off-box\)" on page 29](#).

Note that a Juniper Networks device doesn't consume its computational resources for signature verification.

Install the verified image on your Juniper Networks device. See [Juniper Networks Software Installation and Upgrade Guide](#).

Verify PQC Signatures (Off-Box)

Prerequisites

Ensure you complete the following prerequisites before verifying the PQC signature off the box.

1. Understand the off-box PQC signature verification method. See ["About PQC Signature Verification \(Off-Box\)" on page 27](#).
2. Ensure you have the following files:
 - a. PQC-signed software image
 - b. SHA-512 of the software image
 - c. PQC signature
 - d. Public key associated with the PQC signature
3. Ensure you have the following tools:
 - Install a PQC signature verification tool on your local machine to compare the signature with the hash file. We recommend OpenSSL 3.5 or later.
 - Install a tool such as the sha512sum to generate and validate the SHA-512 checksum of the downloaded software image.

Verification

Perform the following verification steps on your laptop.

1. Fetch the cryptographic hash of the image to verify its SHA-512 checksum for image integrity. Use one of the following commands.
 - Retrieve the hash of the image using the sha512sum tool.

```
user@host:~# sha512sum /data/evo/junos-evo-install-acx-x86-64-25.4R1.8-EVO.iso | cut -d " "  
" -f 1 > image.hash
```

This example uses the sample image *junos-evo-install-acx-x86-64-25.4R1.8-EVO.iso*. Replace it with an image that works for your device.

- Retrieve the hash of the image using the openssl tool.

```
user@host:~# /usr/local/openssl/bin/openssl dgst -sha512 /data/evo/junos-evo-install-acx-x86-64-25.4R1.8-EV0.iso | cut -d " " -f 2 > image.hash
```



TIP: Use cut, awk, or sed to extract the hash. Avoid manual edits to prevent errors.

In this example:

- `junos-evo-install-acx-x86-64-25.4R1.8-EV0.iso` is the name of the PQC-signed software image.
- `image.hash` is the default filename for storing the image hash. You can rename the file.

Compare the SHA-512 checksum in the file with the checksum you received with the downloaded software image. Both checksums must match.

2. Verify the PQC signature of the image.

```
user@host:~# /usr/local/openssl/bin/openssl dgst -verify pubkey.pem -signature junos-evo-install-acx-x86-64-25.4R1.8-EV0.sha512mldsa image.hash
```

In this example:

- `pubkey.pem` is the public key.
- `junos-evo-install-acx-x86-64-25.4R1.8-EV0.sha512mldsa` is the PQC signature of the image.
- `image.hash` is the SHA-512 checksum.

The command displays:

- Verified OK if the verification is successful.
- Verification failure if the verification is invalid. If the signature verification fails:
 - Ensure you're using the appropriate verification tool.
 - Confirm that you have the correct image, signature, and public key.
 - Review the verification steps and ensure that the hash generated from the image matches the hash you received.
 - If the hashes don't match, download the image again and regenerate the hash for verification.
 - If you need help, contact Customer Support.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
25.4R1-EVO	Support introduced for PQC-signed software images with off-box validation in Junos OS Evolved Release 25.4R1.
25.4R1	Support introduced for PQC-signed software images with off-box validation in Junos OS Release 25.4R1.

RELATED DOCUMENTATION

[Junos® OS Software Installation and Upgrade Guide](#)

[Junos® OS Evolved Software Installation and Upgrade Guide](#)

Digital Certificates

SUMMARY

Learn about the digital certificates and find out how to configure digital certificates.

IN THIS SECTION

- [Generate Digital Certificates Manually: Configuration Overview | 32](#)
- [Digital Certificates in Junos OS | 33](#)
- [Generate a Root CA Certificate | 34](#)
- [Manually Generate Self-Signed SSL Certificates | 36](#)
- [Export Certificates to a Specified Location | 37](#)
- [Configure a Root CA Certificate | 37](#)
- [Certificate Chain Implementation | 38](#)
- [Configuring Digital Certificates | 44](#)
- [Requesting a CA Digital Certificate | 46](#)

A digital certificate is an electronic means for verifying your identity through a trusted third party, known as a 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, 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 keypair on the device. See "[Self-Signed Digital Certificates](#)" on page 65.
2. Create a CA profile or profiles containing information specific to a CA. See "[Example: Configure a CA Profile](#)" on page 62.
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 75.
4. Load the certificate onto the device. See "[Example: Load CA and Local Certificates Manually](#)" on page 77.
5. Configure automatic reenrollment. See "[Example: Using SCEP to Automatically Renew a Local Certificate](#)" on page 93.
6. If necessary, load the certificate's CRL on the device. See "[Example: Manually Load a CRL onto the Device](#)" on page 106.

7. If necessary, configure the CA profile with CRL locations. See ["Example: Configure a Certificate Authority Profile with CRL Locations" on page 111](#).

Digital Certificates in Junos OS

For small networks, the use of preshared keys in an IPsec configuration is often sufficient. However, as a network grows, you might find it challenging to add new preshared keys on the local router and all new and existing IPsec peers. The digital certificate implementation helps you scale the IPsec network.

A digital certificate implementation uses the PKI, which requires you to generate a keypair consisting of a public key and a private key. The keys are created with a random number generator and are used to encrypt and decrypt data. In networks that do not use digital certificates, an IPsec-enabled device encrypts data with the private key and IPsec peers decrypt the data with the public key.

With digital certificates you and your IPsec peers request a CA to send you a CA certificate that contains the public key of the CA. Then, you request the CA to enroll a local digital certificate that contains your public key and some additional information. When the CA processes your request, it signs your local certificate with the private key of the CA. You then install the CA certificate and the local certificate in your local router and load the CA certificate in the remote devices before you can establish IPsec tunnels with your peers.

When you request a peering relationship with an IPsec peer, the peer receives a copy of your local certificate. Because the peer already has the CA certificate loaded, it can use the CA's public key contained in the CA certificate to decrypt your local certificate that has been signed by the CA's private key. As a result, the peer now has a copy of your public key. The peer encrypts data with your public key before sending it to you. When your local router receives the data, it decrypts the data with your private key.

In the Junos OS, you must implement the following steps to be able to initially use digital certificates:

- Configure a CA profile to request CA and local digital certificates—The profile contains the name and URL of the CA or registration authority (RA), as well as some retry timer settings.
- Configure certificate revocation list support—A certificate revocation list (CRL) contains a list of certificates canceled before their expiration date. When a participating peer uses a CRL, the CA acquires the most recently issued CRL and checks the signature and validity of a peer's digital certificate. You can request and load CRLs manually, configure an LDAP server to handle CRL processing automatically, or disable CRL processing that is enabled by default.
- Request a digital certificate from the CA—The request can be made either online or manually. Online CA digital certificate requests use the Simple Certificate Enrollment Protocol (SCEP) format. If you request the CA certificate manually, you must also load the certificate manually.

- Generate a private/public key pair—The public key is included in the local digital certificate and the private key is used to decrypt data received from peers.
- Generate and enroll a local digital certificate—The local certificate can be processed online using SCEP or generated manually in the Public-Key Cryptography Standards #10 (PKCS-10) format. If you create the local certificate request manually, you must also load the certificate manually.
- Apply the digital certificate to an IPSec configuration—To activate a local digital certificate, you configure the IKE proposal to use digital certificates instead of preshared keys, reference the local certificate in the IKE policy, and identify the CA in the service set.

Optionally, you can do the following:

- Configure the digital certificate to automatically reenroll—Starting in Junos OS Release 8.5, you can configure automatic reenrollment for digital certificates.
- Monitor digital certificate events and delete certificates and requests—You can issue operational mode commands to monitor IPSec tunnels established using digital certificates and delete certificates or requests.

Generate a Root CA Certificate

To define a self-signed certificate in CLI, you must provide the following details:

- Certificate identifier (generated in the previous step)
- FQDN for the certificate
- E-mail address of the entity owning the certificate
- Common name and the organization involved

Generate a root CA certificate using the Junos OS CLI:

1. From the operational mode, generate a PKI public and private keypair for a local digital certificate.

```
user@host>request security pki generate-key-pair certificate-id certificate-id size size type type
```

Here, you can select the one of the following combinations:

- 1024 bits (RSA/ DSA only)
- 2048 bits (RSA/DSA only)

- 256 bits (ECDSA only)
- 384 bits (ECDSA only)
- 4096 bits (RSA/DSA only)
- 521 bits (ECDSA only)

Example:

```
user@host>request security pki generate-key-pair certificate-id SECURITY-cert size 2048 type
rsa
```

Or

```
user@host>request security pki generate-key-pair certificate-id SECURITY-cert size 521 type
ecdsa
```

2. Define a self-signed certificate.

```
user@host> request security pki local-certificate generate-self-signed certificate-id
certificate-id domain-name domain-name subject subject email email-id add-ca-constraint
```

Example:

```
user@host> request security pki local-certificate generate-self-signed certificate-id
SECURITY-cert domain-name labs.abc.net subject
DC=mydomain.net,L=Sunnyvale,O=Mydomain,OU=LAB,CN=SECURITY email lab@labs.abc.net add-ca-
constraint
```

By configuring the add-ca-constraint option, make sure that the certificate can be used to sign other certificates.

3. From the configuration mode, apply the loaded certificate as root-ca in the SSL proxy profile.

```
[edit]
user@host# set services ssl proxy profile profile-name root-ca certificate-id
```

Example:

```
[edit]
user@host# set services ssl proxy profile SECURITY-SSL-PROXY root-ca SECURITY-cert
```

4. Import the root CA as a trusted CA into client browsers. The root CA certificate is required for the client browsers to trust the certificates signed by the firewall.

Manually Generate Self-Signed SSL Certificates

To manually generate a self-signed SSL certificate on Juniper Networks devices:

1. Establish basic connectivity.
2. If you have root login access, you can manually generate the self-signed certificate by using the following commands:

```
root@host> request security pki generate-key-pair size 512 certificate-id certname
```

```
Generated key pair sslcert, key size 512 bits
```

```
root@host> request security pki local-certificate generate-self-signed certificate-id cert-name email email domain-name domain name ip-address IP address subject "DC= Domain name, CN= Common-Name, OU= Organizational-Unit-name, O= Organization-Name, ST= state, C= Country"
```

```
Self-signed certificate generated and loaded successfully
```

When you generate the certificate, you must specify the subject, e-mail address, and either the domain-name or the IP address.

3. To verify that the certificate was generated and loaded properly, enter the `show security pki local-certificate` operational command and specify `local-certificate` under HTTPS Web management.

```
[edit]
root@host# show system services web-management https local-certificate certname
```

Export Certificates to a Specified Location

When a self-signed certificate is generated using a PKI command, the newly generated certificate is stored in a predefined location (**var/db/certs/common/local**).

Use the following command to export the certificate to a specific location (within the device). You can specify the certificate ID, filename, and type of file format (DER/PEM):

```
user@host> request security pki local-certificate export certificate-id certificate-id filename
filename type der
```

Configure a Root CA Certificate

A CA can issue multiple certificates in the form of a tree structure. A root certificate is the topmost certificate of the tree, the private key of which is used to *sign* other certificates. All certificates immediately below the root certificate inherit the signature or trustworthiness of the root certificate. This is somewhat like the *notarizing* of an identity.

To configure a root CA certificate:

1. Obtain a root CA certificate (by either or importing one)
 - You can generate a root CA certificate using the Junos OS CLI.
 - Obtain a certificate from an external CA (not covered in this topic).
2. Apply the root CA to an SSL proxy profile.

Certificate Chain Implementation

IN THIS SECTION

- [Requirements | 39](#)
- [Overview | 39](#)
- [SSL Certificate: Configuration Overview | 41](#)

The SSL forward proxy supports the certificate chain implementation. Let's discuss certificate chain concepts and how to configure it on an firewall.

- Certificate authority (CA)—CA is a trusted third party responsible for validating the identities of entities (such as websites, email addresses, or companies, or individual persons) and issues a digital certificate by binding cryptographic keys. If your organization owns a CA server, then you become your own CA and use self-signed certificate.
- Root certificate—A Root certificate is a certificate issued by a trusted CA. The root certificate is the topmost certificate of the tree, the private key of which is used to sign other certificates. All certificates immediately below the root certificate inherit the signature or trustworthiness of the root certificate. These certificates are used to establish connection between two endpoints.
- Intermediate CA certificate—An intermediate CA certificate is a subordinate certificate signed by the trusted root specifically to validate an EE certificate.
- Certificate chain—A certificate chain is the ordered list of certificates that contains the SSL certificate, intermediate certificate, and root certificate. Some CAs do not sign with their root certificate, but instead they use an intermediate certificate. An intermediate CA can sign certificates on behalf of the root CA certificate. The root CA signs the intermediate certificate, forming a chain of trust.

The intermediate certificate must be installed on the same server as the SSL certificate so that the connecting devices, such as browsers, applications, and mobile devices can trust the certificate.

When you initiate a connection, the connecting device checks whether the certificate is authentic and is issued by a trusted CA that is embedded in the browser's trusted store.

If the SSL certificate is not from a trusted CA, then the connecting device continues to check if the SSL certificate is issued by an intermediate CA and this intermediate CA is signed by a root CA. The check continues until the device finds the root CA. If the device finds a root CA, a secure connection is established. If the device doesn't find a root CA, then the connection is dropped, and your web browser displays an error message about an invalid certificate or a certificate not trusted.

Certificate Chain Support for SSL Initiation

In the SSL proxy mode, the server sends the complete certificate chain, including the intermediate certificate, to the client for the validation of the certificate chain.

In the non-proxy mode, during the SSL initiation phase, the client sends only the client certificate to the server if requested. Then the server imports the list of trusted CAs to authenticate the client certificate.

During SSL initiation, the devices generate certificate chain and send it along with the client certificate to the server. The certificate chain during the SSL initiation phase enables the server to validate the certificate chain without having to import intermediate CAs of the chain.

The following example shows how to install the certificate chain to enable browsers to trust your certificate.

Requirements

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

Overview

IN THIS SECTION

- [Topology | 39](#)

In this example, you have a domain, example.domain-1, and you want to purchase a certificate from XYZ-Authority for your domain. However, XYZ-Authority is not a root CA and the visiting browser trusts only root-CA certificate. In other words, its certificate is not directly embedded in your browser, and therefore it is not explicitly trusted.

In this case, trust is established in the following manner using the certificate chain (of intermediate certificates).

Topology

Let's try to visualize this chain through [Figure 6 on page 40](#). This figure depicts a full certificate chain, from the root CA certificate to the end-user certificate. The chain terminates at the end-user certificate.

Figure 6: Certification Path from the Certificate Owner to the Root CA

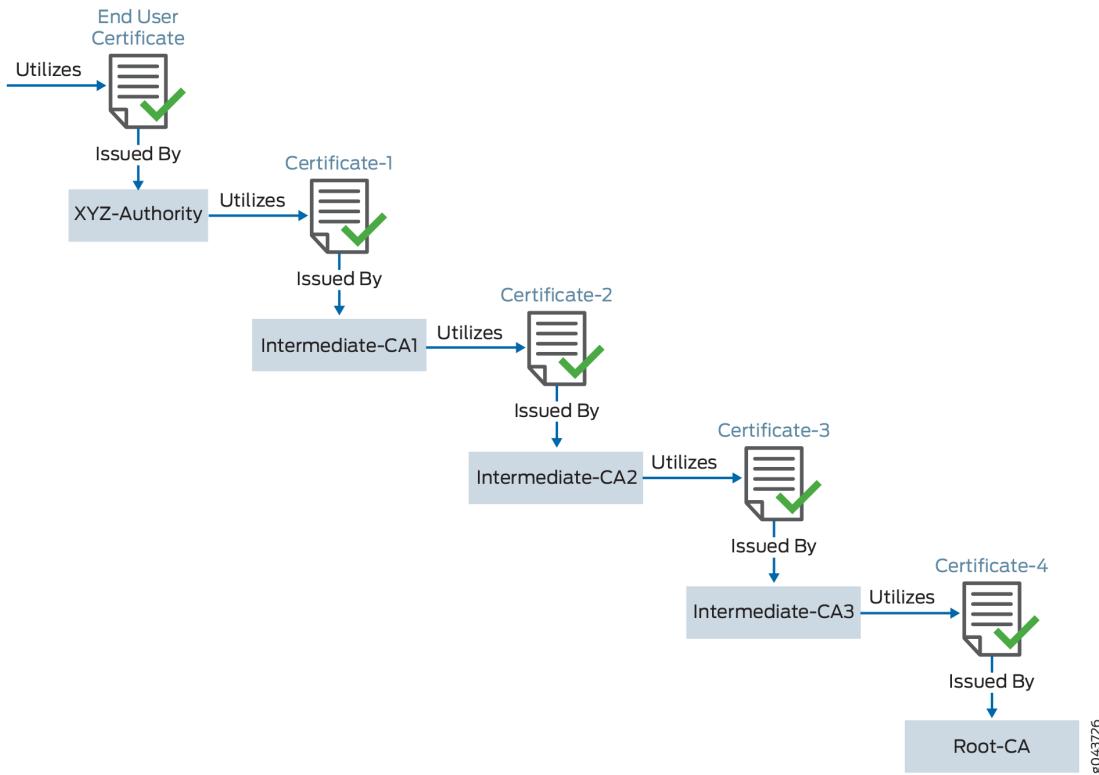


Table 9: Certificate Chain Details

User	Uses Certificate	Signed By	Type
example.domain-1	End-User Certificate	XYZ-Authority	End-User Certificate - the one you purchase from the CA
XYZ-Authority	Certificate-1	Intermediate CA-1	Intermediate Certificate
Intermediate CA-1	Certificate-2	Intermediate CA-2	Intermediate Certificate
Intermediate CA-2	Certificate-3	Intermediate CA-3	Intermediate Certificate

Table 9: Certificate Chain Details (*Continued*)

User	Uses Certificate	Signed By	Type
Intermediate CA-3	Certificate-4	root-example-authority . This is a root CA.	Root Certificate - the one whose certificate is directly embedded in your browser, which the device can therefore explicitly trust

When you install your end-user certificate for the server example.domain-1, you must bundle all the intermediate certificates and install them along with your end-user certificate. The certificate chain includes all the certificates starting from Certificate-1 to Root-CA certificate. Because the web browser trusts the root CA, it also implicitly trusts all the intermediate certificates. If the SSL certificate chain is invalid or broken, your certificate will not be trusted by some devices.

All certificates must be in PEM format.

When you import the concatenated certificate file into the device, the CA provides a bundle of chained certificates that must be added to the signed server certificate. The server certificate must appear before the chained certificates in the combined file.

SSL Certificate: Configuration Overview

IN THIS SECTION

- Configure the Certificate Chain on the Device | [42](#)

To configure the SSL certificate chain:

- Purchase an SSL certificate from a CA that includes a signing certificate and a respective key.
- Configure a trusted CA profile group.
- Load the signing certificate and the key in your device.
- Load the intermediate and root CAs in PKI memory. This certificate file contains all the required CA certificates, one after each other, in the PEM format.
- Create a trusted CA profile for the intermediate or root CA certificate.

- Set up your device to use the signing certificate received from the CA by configuring and applying the SSL proxy profile to a security policy. SSL forward proxy stores this certificate chain information (CA certificate profile name) in the respective SSL profile. As a part of security policy implementation, SSL profiles having the certificate chain information and CA certificates are used.

Certificate chain processing includes the following processes:

- Administrator loads the certificate chain and the local certificate (signing certificate) into the PKI certificate cache.
- The Network Security (nsd) sends a request to the PKI to provide the certificate chain information for a signing certificate configured in the SSL proxy profile.

This example assumes that you have already purchased an SSL certificate from a CA.

Configure the Certificate Chain on the Device

Step-by-Step Procedure

To configure certificate chain:

- Load the local certificate into the PKI memory.

```
user@host>
request security pki local-certificate load filenamessl_proxy_ ca.crt key sslserver.key
certificate-id ssl-inspect-ca
```

The following message is displayed:

```
Local certificate loaded successfully
```

Note that the certificate ID will be used under the root-ca section in the SSL proxy profile.

- Load the intermediate or root CA certificate in the PKI memory.

```
user@host>
request security pki ca-certificate ca-profile-group load ca-group-name ca-latest filename
ca-latest.cert.pem
```

The CA profile includes the certificate information used for authentication. It includes the public key that SSL proxy uses when generating a new certificate.

```
Do you want to load this CA certificate? [yes,no] (no) yes
Loading 1 certificates for group 'ca-latest'.
ca-latest_1: Loading done.
ca-profile-group 'ca-latest' successfully loaded
Success[1] Skipped[0]
```

You can attach this certificate as a certificate chain.

- Attach the CA profile group to the SSL proxy profile. You can attach the trusted CA one at a time or load all in one action.

```
user@host#
set services ssl proxy profile ssl-profile trusted-ca all
```

- Apply the signing certificate as root-ca in the SSL proxy profile.

```
user@host#
set services ssl proxy profile ssl-profile root-ca ssl-inspect-ca
```

- Create a security policy and specify the match criteria for the policy. As match criteria, specify the traffic for which you want to enable SSL proxy. This example assumes that you have already created security zones based on the requirements.

```
user@host#
set security policies from-zone trust to-zone untrust policy 1 match source-address any
user@host#
set security policies from-zone trust to-zone untrust policy 1 match destination-address any
user@host#
set security policies from-zone trust to-zone untrust policy 1 match application any
user@host#
set security policies from-zone trust to-zone untrust policy 1 then permit application-
services ssl-proxy profile-name ssl-profile
```

SSL forward proxy stores this certificate chain information (CA certificate profile name) into respective the SSL profile. As a part of security policy implementation, SSL profiles having the certificate chain information and CA certificates are used.

You can view the certificate chain on the connecting browser, that is, the client.

Configuring Digital Certificates

IN THIS SECTION

- [Digital Certificates Overview | 44](#)
- [Obtaining a Certificate from a CA for an ES PIC | 45](#)
- [Request a CA Digital Certificate | 45](#)
- [Generate a Private and Public Keypair for Digital Certificates for an ES PIC | 46](#)

Digital Certificates Overview

A *digital certificate* provides a way of authenticating users through a trusted third-party called a CA. The CA validates the identity of a certificate holder and “signs” the certificate to attest that it has not been forged or altered.

A certificate includes the following information:

- The DN of the owner. A DN is a unique identifier and consists of a fully qualified name including the common name (CN) of the owner, the owner’s organization, and other distinguishing information.
- The public key of the owner.
- The date on which the certificate was issued.
- The date on which the certificate expires.
- The DN of the issuing CA.
- The digital signature of the issuing CA.

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

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

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



NOTE: Certificates without an alternate subject name are not appropriate for IPsec services.

Obtaining a Certificate from a CA for an ES PIC

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

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



NOTE: For the dynamic registration of digital certificates, the Junos OS supports only the SCEP.

Request a CA Digital Certificate

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

```
user@host> request security certificate enroll filename ca_verisign ca-file verisign ca-name
xyzcompany url
http://hostname/path/filename
URL: http://hostname/path/filename name: example.com CA file: verisign Encoding: binary
Certificate enrollment has started. To see the certificate enrollment status, check the key
management process (kmd) log file at /var/log/kmd. <-----
```



NOTE: Initially, each router is initially manually enrolled with a CA.

Generate a Private and Public Keypair for Digital Certificates for an ES PIC

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

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

name specifies the filename in which the device stores the public and private keys.

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

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



NOTE: When you use *SCEP*, the Junos OS only supports RSA.

The following example shows how to generate a private and public keypair:

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

Requesting a CA Digital Certificate

IN THIS SECTION

- [Request a CA Digital Certificate | 46](#)
- [Generate a Public–Private Keypair | 47](#)
- [Generate and Enroll a Local Digital Certificate | 47](#)
- [Apply Local Digital Certificate to an IPsec Configuration | 47](#)
- [Configure Automatic Reenrollment of Digital Certificates | 48](#)

Request a CA Digital Certificate

You can request a CA digital certificate either online or manually. To request a digital certificate online from a CA or a RA by using SCEP, issue the `request security pki ca-certificate enroll ca-profile-name` command.

If you had obtained the CA digital certificate manually through e-mail or other OOB mechanism, you must load it manually. To manually install a certificate in your router, issue the request security pki ca-certificate load ca-profile *profile_name* filename */path/filename.cert* command.

Generate a Public-Private Keypair

A keypair 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. To generate a private-public keypair, issue the request security pki generate-key-pair certificate-id *certificate-id-name* command.

Generate and Enroll a Local Digital Certificate

You can generate and enroll a local digital certificate either online or manually. To generate and enroll a local certificate online by using SCEP, issue the request security pki local-certificate enroll command. To generate a local certificate request manually in the PKCS-10 format, issue the request security pki generate-certificate-request command.

If you create the local certificate request manually, you must also load the certificate manually. To manually install a certificate in your router, issue the request security pki local-certificate load command.

Apply Local Digital Certificate to an IPsec Configuration

To activate a local digital certificate, configure the IKE proposal to use digital certificates instead of preshared keys, reference the local certificate in the IKE policy, and identify the CA or RA in the service set. To enable the IKE proposal for digital certificates, include the rsa-signatures statement at the [edit services ipsec-vpn ike proposal *proposal-name* authentication-method] hierarchy level. To reference the local certificate in the IKE policy, include the local-certificate statement at the [edit services ipsec-vpn ike policy *policy-name*] hierarchy level. To identify the CA or RA in the service set, include the trusted-ca statement at the [edit services service-set *service-set-name* ipsec-vpn-options] hierarchy level.

```
[edit services]
service-set service-set-name {
    .....
    ipsec-vpn-options {
        trusted-ca ca-profile-name;
    }
}
ipsec-vpn {
    ike {
        proposal proposal-name {
            .....
            authentication-method [pre-shared-keys | rsa-signatures];
        }
    }
}
```

```

policy policy-name {
    ....
    local-certificate certificate-id-name;
}
}

```

Configure Automatic Reenrollment of Digital Certificates

You can configure automatic reenrollment for digital certificates. This feature is not enabled by default. To configure automatic reenrollment for digital certificates, include the auto-re-enrollment statement at the [edit security pki] hierarchy level:

```

[edit]
security {
    pki {
        auto-re-enrollment {
            certificate-id certificate-name {
                ca-profile ca-profile-name;
                challenge-password password;
                re-enroll-trigger-time-percentage percentage; # Percentage of validity-period
                # (specified in certificate) when automatic
                # reenrollment should be initiated.
                re-generate-keypair;
                validity-period number-of-days;
            }
        }
    }
}

```

Configure Digital Certificates for an ES PIC

IN THIS SECTION

- [Configure the CA Properties for an ES PIC | 50](#)
- [Configure the Cache Size | 53](#)

- Configure the Negative Cache | [53](#)
- Configure the Number of Enrollment Retries | [54](#)
- Configure the Maximum Number of Peer Certificates | [54](#)
- Configure the Path Length for the Certificate Hierarchy | [54](#)

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

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

```
[edit security]
certificates {
    cache-size bytes;
    cache-timeout-negative seconds;
    certification-authority ca-profile-name {
        ca-name ca-identity;
        crl filename;
        encoding (binary | pem);
        enrollment-url url-name;
        file certificate-filename;
        ldap-url url-name;
    }
    enrollment-retry attempts;
    local certificate-filename {
        certificate-key-string;
        load-key-file URL key-file-name;
    }
    maximum-certificates number;
    path-length certificate-path-length;
}
ike {
    policy ike-peer-address {
        description policy;
        encoding (binary | pem);
        identity identity-name;
        local-certificate certificate-filename;
    }
}
```

```

local-key-pair private-public-key-file;
mode (aggressive | main);
pre-shared-key (ascii-text key | hexadecimal key);
proposals [ proposal-names ];
}
}

```

Tasks to configure digital certificates for ES PICs are:

Configure the CA Properties for an ES PIC

IN THIS SECTION

- [Specify the CA Name | 51](#)
- [Configure the CRL | 51](#)
- [Configure the Type of Encoding Your CA Supports | 51](#)
- [Specify an Enrollment URL | 52](#)
- [Specify a File to Read the Digital Certificate | 52](#)
- [Specify an LDAP URL | 52](#)

To configure a CA and its properties for an ES PIC, include the following statements at the [edit security certificates] hierarchy level:

```

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

```

ca-profile-name is the CA profile name.

Perform the following tasks to configure the CA properties:

Specify the CA Name

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

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

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

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

Configure the CRL

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

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

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

Configure the Type of Encoding Your CA Supports

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

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

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

Specify an Enrollment URL

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

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

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

Specify a File to Read the Digital Certificate

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

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

Specify an LDAP URL

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

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

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

Configure the Cache Size

By default, the cache size is 2 MB. To configure total cache size for digital certificates, include the cache-size statement at the [edit security certificates] hierarchy level:

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

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



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

Configure the Negative Cache

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

By default, the negative cache is 20 seconds. To configure the negative cache, include the cache-timeout-negative statement at the [edit security certificates] hierarchy level:

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

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



NOTE: Configuring a large negative cache value can make you susceptible to a DoS attack.

Configure the Number of Enrollment Retries

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

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

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

Configure the Maximum Number of Peer Certificates

By default, the maximum number of peer certificates to be cached is 1024. To configure the maximum number of peer certificates to be cached, include the maximum-certificates statement at the [edit security certificates] hierarchy statement level:

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

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

Configure the Path Length for the Certificate Hierarchy

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

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

By default, the maximum certificate path length is set to 15. The root anchor is 1.

To configure path length, include the path-length statement at the [edit security certificates] hierarchy level:

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

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

Certificate Authority

SUMMARY

Understand how to manage CA.

IN THIS SECTION

- [Certificate Authority Profiles | 55](#)
- [Configure CA Profiles | 56](#)
- [Configure a Trusted CA Group | 58](#)
- [Example: Configure a CA Profile | 62](#)
- [Example: Configure an IPv6 Address as the Source Address for a CA Profile | 64](#)

A CA profile defines 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.

Certificate Authority Profiles

A CA profile configuration contains information specific to a CA. You can have multiple CA profiles on 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 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, you can validate a configured IKE peer with a specified CA server or group of CA servers. You can create a group of trusted CA servers with the trusted-ca-group configuration statement at the [edit security pki] hierarchy level; you can specify one or multiple CA profile. The trusted CA server is bound to the IKE policy configuration for the peer at [edit security ike policy *policy* certificate] hierarchy level.

If you configure the proxy profile in the CA profile, the device connects to the proxy host instead of the CA server during certificate enrollment, verification, or revocation. The proxy host communicates with the CA server with the requests from the device, and then relays the response to the device.

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

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

Configure CA Profiles

A CA profile configuration contains information specific to a CA. You can have multiple CA profiles on 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 CA profile (for example, Microsoft-2008). You can group multiple CA profiles in one trusted CA group for a given topology.

In the following example, you create a CA profile called ca-profile-security with CA identity microsoft-2008. You then create proxy profile to the CA profile.

1. From the configuration mode, configure the CA profile used for loading the certificate.

```
[edit]
```

```
user@host# set security pki ca-profile profile-name ca-identity
```

ca-identity

Example:

```
user@host# set security pki ca-profile ca-profile-security ca-identity example.com
```

2. Commit the configuration.

[edit]

```
user@host# commit
```

3. From the operational mode, load the certificate using PKI commands.

```
user@host> request security pki ca-certificate load ca-profile profile-name filename filename
```

Example:

```
user@host> request security pki ca-certificate load ca-profile ca-profile-security filename
srx-123.crt
```

4. From the configuration mode, disable the revocation check (if required).

[edit]

```
user@host# set security pki ca-profile profile-name ca-identity ca-identity revocation-check
disable
```

Example:

[edit]

```
user@host# set security pki ca-profile ca-profile-security ca-
identity example.com revocation-check disable
```

5. From the configuration mode, configure the loaded certificate as a trusted CA in the SSL proxy profile. You can configure more than one trusted CA for a profile.

```
[edit]
user@host# set services ssl proxy profile ssl-proxy-profile-name
trusted-ca ca-profile-name
```

Example:

```
[edit]
user@host# set services ssl proxy profile ssl-proxy-sample
trusted-ca ca-profile-security
```

6. (Optional) If you have multiple trusted CA certificates, you do not have to specify each trusted CA separately. You can load *all* the trusted CA certificates using the following command from configuration mode. Alternatively, you can import a set of trusted CAs from your browser into the Firewall.

```
[edit]
user@host# set services ssl proxy profile ssl-proxy-profile-name
root-ca ssl-inspect-ca
user@host# set services ssl proxy profile ssl-proxy-profile-name
trusted-ca all
```

Configure a Trusted CA Group

IN THIS SECTION

- [Create a Trusted CA Group | 59](#)
- [Delete a CA Profile from a Trusted CA Group | 60](#)
- [Delete a Trusted CA Group | 61](#)

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

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 belong 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.

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 58 topic:

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 58](#) 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.

Example: Configure a CA Profile

IN THIS SECTION

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

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 retries 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 | [63](#)

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. 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
```

3. 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
```

4. Specify the enrollment retry value.

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

5. 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
```

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

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

Proxy Authentication Support

Define proxy profiles at the system level to manage authenticated outbound connections. You can configure these profiles under the [edit services proxy] hierarchy, where multiple proxy profiles can be created. Each CA profile can reference at most one proxy profile.

- Define proxy server's host and port settings.

```
[edit]
user@host# set services proxy profile px-profile protocol http host x.x.x.x port xxxx
```

- Configure proxy authentication within the proxy profile. By setting a username and password, you can ensure secure access to external feeds and services.

```
[edit]
user@host# set services proxy profile px-profile protocol http username <username>
user@host# set services proxy profile px-profile protocol http password <password>
```

- Attach the proxy profile to the CA profile.

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

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 a self-signed digital certificate.

IN THIS SECTION

- [Self-Signed Certificates | 66](#)
- [Example: Generate a Public-Private Keypair | 67](#)

- [Manually Generate Self-Signed SSL Certificates | 68](#)
- [Example: Manually Generate Self-Signed Certificates | 69](#)
- [Manage Automatically Generated Self-Signed Certificates | 70](#)
- [Enable HTTPS and XNM-SSL Services on Switches Using Self-Signed Certificates \(CLI Procedure\) | 71](#)

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 CA.

Self-signed certificates allow for use of SSL based services without requiring 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, because a client cannot verify that the server connected to is the one advertised in the certificate. Self-signed certificates are valid for five years.

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

- Automatic generation-The Juniper Networks device creates the certificate automatically. An automatically generated self-signed certificate is configured on the device by default. After you initialize the device, it checks for the presence of an automatically generated self-signed certificate. If the device does not find one, it generates one and saves it in the file system.

A self-signed certificate that is automatically generated by the device is similar to an 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.

- Manual generation-You create the self-signed certificate for the Juniper Networks 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.

A manually generated self-signed certificate is one example of a 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 Keypair

IN THIS SECTION

- Requirements | [67](#)
- Overview | [67](#)
- Configuration | [67](#)
- Verification | [68](#)

This example shows how to generate a public-private keypair.

Requirements

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

Overview

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

Configuration

IN THIS SECTION

- Procedure | [67](#)

Procedure

Step-by-Step Procedure

To generate a public-private keypair:

- Create a certificate keypair.

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

Verification

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

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

Manually Generate Self-Signed SSL Certificates

To manually generate a self-signed SSL certificate on Juniper Networks devices:

1. Establish basic connectivity.
2. If you have root login access, you can manually generate the self-signed certificate by using the following commands:

```
root@host> request security pki generate-key-pair size 512 certificate-id certname
```

```
Generated key pair sslcert, key size 512 bits
```

```
root@host> request security pki local-certificate generate-self-signed certificate-id cert-name email email domain-name domain name ip-address IP address subject "DC= Domain name, CN= Common-Name, OU= Organizational-Unit-name, O= Organization-Name, ST= state, C= Country"
```

```
Self-signed certificate generated and loaded successfully
```

When you generate the certificate, you must specify the subject, e-mail address, and either the domain-name or the IP address.

3. To verify that the certificate was generated and loaded properly, enter the show security pki local-certificate operational command and specify local-certificate under HTTPS Web management.

```
[edit]
```

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

Example: Manually Generate Self-Signed Certificates

IN THIS SECTION

- Requirements | [69](#)
- Overview | [69](#)
- Configuration | [69](#)
- Verification | [70](#)

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

Requirements

Before you begin, generate a public private keypair. See ["Digital Certificates" on page 31](#).

Overview

For a manually generated self-signed certificate, you specify the 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 | [70](#)

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 172.16.3.4 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
```

Note 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
```

Manage Automatically Generated Self-Signed Certificates

After you initialize the device, 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
```

Note the Certificate identifier details in the output. It displays the following details 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.

Enable HTTPS and XNM-SSL Services on Switches Using Self-Signed Certificates (CLI Procedure)

You can use the system-generated self-signed certificate or a manually generated self-signed certificate to enable Web management HTTPS and XNM-SSL services on a switch.

- Use the following command to enable HTTPS services using the automatically generated self-signed certificate:

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

- Use the following command to enable HTTPS services using a manually generated self-signed certificate:

```
[edit]
user@switch# set system services web-management https pki-local-certificate certificate-id-name
```

The value of the *certificate-id-name* must match the name you specified when you generated the self-signed certificate manually.

- To enable XNM-SSL services using a manually generated self-signed certificate, use the following command:

```
[edit]
user@switch# set system services xnm-ssl local-certificate certificate-id-name
```

The value of the *certificate-id-name* must match the name you specified when you generated the self-signed certificate manually.

Enroll a Certificate

SUMMARY

Learn about how to enroll and manage various PKI certificates.

IN THIS SECTION

- [Enroll Digital Certificates Online: Configuration Overview | 73](#)
- [Certificate Enrollment | 73](#)
- [Enroll a CA Certificate Using SCEP | 90](#)
- [Enroll a CA Certificate Online Using CMPv2 | 96](#)
- [Install a Digital Certificate on Your Router | 98](#)
- [Understand ACME Protocol | 102](#)
- [Platform-Specific SSL Termination Services Behavior | 105](#)

A 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 SCEP.

Enroll Digital Certificates Online: Configuration Overview

You can use either CMPv2 or SCEP to enroll digital certificates. To enroll a certificate online:

1. Generate a keypair on the device. See ["Self-Signed Digital Certificates" on page 65](#).
2. Create a CA profile or profiles containing information specific to a CA. See ["Example: Configure a CA Profile" on page 62](#).
3. For SCEP only, enroll the CA certificate. See ["Enroll a CA Certificate Online Using SCEP" on page 90](#).
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 91](#).
5. Configure automatic reenrollment. See ["Example: Using SCEP to Automatically Renew a Local Certificate" on page 93](#).

Certificate Enrollment

IN THIS SECTION

- [Online CA Certificate Enrollment | 74](#)
- [Local Certificate Requests | 74](#)
- [CMPv2 and SCEP Certificate Enrollment | 74](#)
- [Example: Manually Generate a CSR for the Local Certificate | 75](#)
- [Example: Manually Load CA and Local Certificates | 77](#)
- [Configure PKI and SSL Forward Proxy to Authenticate Users | 79](#)
- [Configure Digital Certificates for Adaptive Services Interfaces | 80](#)

Online CA Certificate Enrollment

With SCEP, you can configure your Juniper Networks device to obtain a 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 EE certificate in PKCS #10 format from a keypair 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 name (O), locality (L), state (ST), country (C), and domain component (DC). Additionally, a SAN is associated in the following form:

- IP address
- E-mail address
- FQDN

Specify the subject name in the DN 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 you configure the device as a dynamic peer. Instead, you can use an FQDN (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.

CMPv2 and SCEP Certificate Enrollment

Based on your deployment environment, you can use either CMPv2 or SCEP for online certificate enrollment. This topic describes some of the basic differences between the two protocols.

[Table 10 on page 74](#) describes the differences between the CMPv2 and SCEP certificate enrollment protocols.

Table 10: CMPv2 vs SCEP Certificate Enrollment

Attribute	CMPv2	SCEP
Supported certificate types:	DSA, ECDSA, and RSA	RSA only

Table 10: CMPv2 vs SCEP Certificate Enrollment (*Continued*)

Attribute	CMPv2	SCEP
Supported standards	RFCs 4210 and 4211	ietf draft

Certificate enrollment, reenrollment requests, and responses differ between CMPv2 and SCEP. With CMPv2, 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.

Example: Manually Generate a CSR for the Local Certificate

IN THIS SECTION

- [Requirements | 75](#)
- [Overview | 75](#)
- [Configuration | 76](#)
- [Verification | 76](#)

This example shows how to generate a CSR manually.

Requirements

Generate a public and private key. See ["Self-Signed Digital Certificates" on page 65](#).

Overview

In this example, you generate a certificate request using the certificate ID of a public-private keypair 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 are also displayed.

Configuration

IN THIS SECTION

- [Procedure | 76](#)

Procedure

Step-by-Step Procedure

To generate a local certificate manually:

- Specify the 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 CSR, 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: Manually Load CA and Local Certificates

IN THIS SECTION

- Requirements | [77](#)
- Overview | [77](#)
- Configuration | [78](#)
- Verification | [78](#)

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

Requirements

Before you begin:

- Generate a public-private keypair. See "[Self-Signed Digital Certificates](#)" on page [65](#).
- Create a CA profile. See "[Certificate Authority](#)" on page [55](#)

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 [75](#).

Overview

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

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

Then, you can load the following certificate files onto a device running Junos OS:

- A local or 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 certificate onto the device.

Configuration

IN THIS SECTION

- [Procedure | 78](#)

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
```

Configure PKI and SSL Forward Proxy to Authenticate Users

Non-domain users can configure PKI to validate integrity, confidentiality, and authenticity of traffic. PKI includes digital certificates issued by the CA, certificate validity and expiration dates, details about the certificate owner and issuer, and security policies.

For any non-domain user or domain user on a non-domain machine, the administrator specifies a captive portal to force the user to perform firewall authentication (if the firewall supports captive portal for the traffic type). After the user enters a name and password and passes firewall authentication, the device gets firewall authentication user or group information and can enforce the user firewall policy to control the user accordingly. In addition to captive portal, if the IP address or user information is not available from the event log, the user can again log in to the Windows PC to generate an event log entry. Then the system generates the user's authentication entry accordingly.

To enable the device to authenticate the users through HTTPS, the users must configure and enable the SSL forward proxy. You need to generate a local certificate, add an SSL termination profile, add an SSL proxy profile, and reference the SSL proxy profile in the security policy. If the SSL forward proxy is not enabled, the firewall cannot authenticate users who are using HTTPS, but for users who are using HTTP, FPT, and Telnet, the authentication can be performed as expected.

To generate PKI and enable SSL forward proxy:

1. Generate a PKI public-private keypair for a local digital certificate.

```
user@host# request security pki generate-key-pair certificate-id ssl-inspect-ca size 2048
type rsa
```

2. Manually generate a self-signed certificate for the given DN.

```
user@host# request security pki local-certificate generate-self-signed certificate-id ssl-
inspect-ca domain-name www.mycompany.net subject "CN=www.mycompany.com,OU=IT,O=MY
COMPANY,L=Sunnyvale,ST=CA,C=US" email security-admin@mycompany.net
```

3. Define the access profile to be used for SSL termination services.

```
user@host# set services ssl termination profile for_userfw server-certificate ssl-inspect-ca
```

4. Configure the loaded certificate as root-ca in the SSL proxy profile.

```
user@host# set services ssl proxy profile ssl-inspect-profile root-ca ssl-inspect-ca
```

5. Specify the `ignore-server-auth-failure` option if you do not want to import the entire CA list and you do not want dropped sessions.

```
user@host# set services ssl proxy profile ssl-inspect-profile actions ignore-server-auth-failure
```

6. Add an SSL termination profile into security policies.

```
user@host# set security policies from-zone untrust to-zone trust policy p1 then permit
firewall-authentication user-firewall ssl-termination-profile for_userfw
```

7. Commit the configuration.

```
user@host# commit
```

Use [Feature Explorer](#) to confirm platform and release support for specific features.

Review the ["Platform-Specific SSL Termination Services Behavior" on page 105](#) section for notes related to your platform.

Configure Digital Certificates for Adaptive Services Interfaces

IN THIS SECTION

- Configure the Certificate Authority Properties | [81](#)
- Configure the Certificate Revocation List | [83](#)
- Manage Digital Certificates | [84](#)
- Configure Auto-Reenrollment of a Router Certificate | [87](#)

A *digital certificate* implementation uses the PKI, which requires that you generate a keypair consisting of a public key and a private key. The keys are created with a random number generator and are used to encrypt and decrypt data. In networks that do not use digital certificates, an *IPsec*-enabled device encrypts data with the private key and IPsec peers decrypt the data with the public key.

With digital certificates, the key sharing process requires an additional level of complexity. First, you and your IPsec peers request that a *CA* send you a CA certificate that contains the public key of the CA. Next you request the CA to assign you a local digital certificate that contains the public key and some

additional information. When the CA processes your request, it signs your local certificate with the private key of the CA. Then you install the CA certificate and the local certificate in your router and load the CA in remote devices before you can establish IPsec tunnels with your peers.

For digital certificates, the Junos OS supports VeriSign, Entrust, Cisco Systems, and Microsoft Windows CAs for the Adaptive Services (AS) and Multiservices PICs.

The following tasks enable you to implement digital certificates.

Configure the Certificate Authority Properties

IN THIS SECTION

- [Specify the CA Profile Name | 81](#)
- [Specify an Enrollment URL | 82](#)
- [Specify the Enrollment Properties | 82](#)

A CA is a trusted third-party organization that creates, enrolls, validates, and revokes digital certificates.

To configure a CA and its properties for the AS and Multiservices PICs, include the following statements at the [edit security pki] hierarchy level:

```
[edit security pki]
ca-profile ca-profile-name {
    ca-identity ca-identity;
    enrollment {
        url url-name;
        retry number-of-attempts;
        retry-interval seconds;
    }
}
```

Tasks for configuring the CA properties are:

Specify the CA Profile Name

The CA profile contains the name and URL of the CA or RA, as well as some retry-timer settings. CA certificates issued by Entrust, VeriSign, Cisco Systems, and Microsoft are compatible.

To specify the CA profile name, include the `ca-profile` statement at the `[edit security pki]` security level:

```
[edit security pki]
ca-profile ca-profile-name;
```

You also need to specify the name of the CA identity used in the certificate request. This name is typically the domain name. To specify the name of the CA identity, include the `ca-identity` statement at the `[edit security pki ca-profile ca-profile-name]` level:

```
[edit security pki ca-profile ca-profile-name]
ca-identity ca-identity;
```

Specify an Enrollment URL

You specify the CA location where your router should send the SCEP-based certificate enrollment requests. To specify the CA location by naming the CA URL, include the `url` statement at the `[edit security pki enrollment]` hierarchy level:

```
[edit security pki ca-profile ca-profile-name enrollment]
url url-name;
```

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

Specify the Enrollment Properties

You can specify the number of times a router will be re-sent a certificate request and the amount of time, in seconds, the router should wait between enrollment attempts.

By default, the number of enrollment retries is set to 0, an infinite number of retries. To specify how many times a router will be re-sent a certificate request, include the `retry number-of-attempts` statement at the `[edit security pki ca-profile ca-profile-name enrollment]` hierarchy level:

```
[edit security pki ca-profile ca-profile-name enrollment]
retry number-of-attempts;
```

The range for *number-of-attempts* is from 0 through 100.

To specify the amount of time, in seconds, that a router should wait between enrollment attempts, include the `retry-interval seconds` statement at the `[edit security pki ca-profile ca-profile-name enrollment]` hierarchy level:

```
[edit security pki ca-profile ca-profile-name enrollment]
  retry-interval seconds;
```

The range for *seconds* is from 0 through 3600.

Configure the Certificate Revocation List

IN THIS SECTION

- [Specify an LDAP URL | 83](#)
- [Configure the Interval Between CRL Updates | 84](#)
- [Override Certificate Verification if CRL Download Fails | 84](#)

Tasks to configure the CRL is:

Specify an LDAP URL

You can specify the URL for the LDAP server where your CA stores its current CRL. If the CA includes the Certificate Distribution Point (*CDP*) in the digital certificate, you do not need to specify a URL for the LDAP server. The CDP is a field within the certificate that contains information about how to retrieve the CRL for the certificate. The router uses this information to download the CRL automatically.

Configure an LDAP URL if you want to use a different CDP from the one specified in the certificate. Any LDAP URL you configure takes precedence over the CDP included in the certificate.

You can configure up to three URLs for each CA profile.

If the LDAP server requires a password to access the CRL, you need to include the `password` statement.

To configure the router to retrieve the CRL from the LDAP server, include the `url` statement and specify the URL name at the `[edit security pki ca-profile ca-profile-name revocation-check crl]` hierarchy level:

```
[edit security pki ca-profile ca-profile-name revocation-check crl]
  url {
```

```
url-name;
}
```

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

To specify to use a password to access the CRL, include the password statement at the [edit security pki ca-profile *ca-profile-name* revocation-check crl url] hierarchy level:

```
[edit security pki ca-profile ca-profile-name revocation-check crl url]
password password;
```

password is the secret password that the LDAP server requires for access.

Configure the Interval Between CRL Updates

By default, the time interval between CRL updates is 24 hours. To configure the amount of time between CRL updates, include the refresh-interval statement at the [edit security pki ca-profile *ca-profile-name* revocation-check crl] hierarchy level:

```
[edit security pki ca-profile ca-profile-name revocation-check crl]
refresh-interval number-of-hours;
```

The range for number of hours is from 0 through 8784.

Override Certificate Verification if CRL Download Fails

By default, if the router either cannot access the LDAP URL or retrieve a valid CRL, certificate verification fails and the IPsec tunnel is not established. To override this behavior and permit the authentication of the IPsec peer when the CRL is not downloaded, include the disable on-download-failure statement at the [edit security pki ca-profile *ca-profile-name* revocation-check crl] hierarchy level:

```
[edit security pki ca-profile ca-profile-name revocation-check crl]
disable on-download-failure;
```

Manage Digital Certificates

IN THIS SECTION

- Request a CA Digital Certificate for AS and Multiservices PICs | 85

- [Generate a Public/Private Keypair | 85](#)
- [Generate and Enroll a Local Digital Certificate | 86](#)

After you configure the CA profile, you can request a CA certificate from the trusted CA. Next, you must generate a public/private keypair. When the keypair is available, you can generate a local certificate either online or manually.

Tasks to manage digital certificates are:

Request a CA Digital Certificate for AS and Multiservices PICs

For information about how to configure a CA profile, see ["Configure the Certificate Authority Properties" on page 81](#).

In this example, the certificate is enrolled online and installed into the router automatically.

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

```
Received following certificates:
Certificate: C=us, O=juniper
  Fingerprint:00:8e:6f:58:dd:68:bf:25:0a:e3:f9:17:70:d6:61:f3:53:a7:79:10
Certificate: C=us, O=juniper, CN=First Officer
  Fingerprint:bc:78:87:9b:a7:91:13:20:71:db:ac:b5:56:71:42:ad:1a:b6:46:17
Certificate: C=us, O=juniper, CN=First Officer
  Fingerprint:46:71:15:34:f0:a6:41:76:65:81:33:4f:68:47:c4:df:78:b8:e3:3f
Do you want to load the above CA certificate ? [yes,no] (no) yes
```

If you obtain the CA certificate directly from the CA (for example, as an e-mail attachment or Web site download), you can install it with the `request security pki ca-certificate load` command. For more information, see the [CLI Explorer](#).

Generate a Public/Private Keypair

After obtaining a certificate for an AS PIC or MS-PIC, you must generate a public-private key before you can generate a local certificate. The public key is included in the local digital certificate and the private key is used to decrypt data received from peers. To generate a public-private keypair, issue the `request security pki generate-key-pair certificate-id certificate-id-name` command.

The following example shows how to generate a public-private key for an AS PIC or MS-PIC:

```
user@host>request security pki generate-key-pair certificate-id local-trust2
Generated key pair local-trust2, key size 1024 bits
```

Generate and Enroll a Local Digital Certificate

You can generate and enroll local digital certificates either online or manually. To generate and enroll a local certificate online by using the SCEP for an AS PIC or MS-PIC, issue the request security pki local-certificate enroll command. To generate a local certificate request manually in the PKCS-10 format, issue the request security pki generate-certificate-request command.

If you create the local certificate request manually, you must also load the certificate manually. To manually install a certificate in your router, issue the request security pki local-certificate load command.

The following example shows how to generate a local certificate request manually and send it to the CA for processing:

```
user@host> request security pki generate-certificate-request certificate-id local-trust2
domain-name router2.example.com filename entrust-req2
subject cn=router2.example.com
```

```
Generated certificate request
-----BEGIN CERTIFICATE REQUEST-----
MIIBoTCCAQoCAQAwGjEYMBYGA1UEAxMPdHAxLmp1bmlwZXIubmV0MIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQCiUFklQws1Ud+AqN5DDxRs2kVyKEhh9qoVFnz+
Hz4c9vsy3B8ElwTJ1kmIt2cB3yifB6zePd+6WYpf57Crwre7YqPkiXM31F6z3YjX
H+1BPNbCxNWYvyrnSyVYDbFj8o0Xyqog8ACDfVL2JBWrPNBYy7imq/K9soDBbAs6
5hZqqwIDAQABoEcwRQYJKoZIhvcNAQkOMTgwnjAOBgNVHQ8BAf8EBAMCB4AwJAYD
VR0RAQH/BBowGIIWdHAxLmVuZ2xhYi5qdW5pcGVyLm5ldDANBgkqhkiG9w0BAQQF
AAOBgQBc2rq1v5S0QXH7LCb/FdqAL8ZM6GoaN5d6cGwq4bB6a7UQFgtoH406gQ3G
3iH0Zfz4xMIBpJYuGd1dkqgvcDoH3AgTsLkfn7Wi3x5H2qeQVs9bvL4P5nvEZLND
EIMUHwteolZCiZ70f09Fer9cXWHSQs1UtXtgPqQJy2xIeImLgw==
-----END CERTIFICATE REQUEST-----
Fingerprint:
0d:90:b8:d2:56:74:fc:84:59:62:b9:78:71:9c:e4:9c:54:ba:16:97 (sha1)
1b:08:d4:f7:90:f1:c4:39:08:c9:de:76:00:86:62:b8 (md5)
```

The trusted CA digitally signs the local certificate and returns it to you. Copy the certificate file into the router and load the certificate:

```
user@host> request security pki local-certificate load filename /tmp/router2-cert certificate-id
local-entrust2
Local certificate local-entrust2 loaded successfully
```

The name of the file sent to you by the CA might not match the name of the certificate identifier. However, the certificate-id name must always match the name of the keypair you generated for the router.

After the local and CA certificates have been loaded, you can reference certificates in your IPsec configuration. Using default values in the AS and Multiservices PICs, you do not need to configure an IPsec proposal or an IPsec policy. However, you must configure an IKE proposal that specifies the use of digital certificates, reference the IKE proposal and locate the certificate in an IKE policy, and apply the CA profile to the service set.

Configure Auto-Reenrollment of a Router Certificate

IN THIS SECTION

- [Specify the Certificate ID | 88](#)
- [Specify the CA Profile | 89](#)
- [Specify the Challenge Password | 89](#)
- [Specify the Reenroll Trigger Time | 89](#)
- [Specify the Regenerate Keypair | 89](#)
- [Specify the Validity Period | 90](#)

Use the auto-re-enrollment statement to configure automatic reenrollment of a specified existing router certificate before its existing expiration date. This function automatically reenrolls the router certificate. The reenrollment process requests the CA to issue a new router certificate with a new expiration date. The date of auto-reenrollment is determined by the following parameters:

- `re-enroll-trigger-time`—The percentage of the difference between the router certificate start date/time (when the certificate was generated) and the validity period. The `re-enroll-trigger-time` used to specify how long auto-reenrollment should be initiated before expiration.

- **validity-period**—The number of days after issuance when the router certificate will expire, as set when a certificate is generated.

By default, this feature is not enabled unless configured explicitly. This means that a certificate that does not have auto-reenrollment configured will expire on its normal expiration date.

The **ca-profile** statement specifies which CA will be contacted to reenroll the expiring certificate. This is the CA that issued the original router certificate.

The **challenge-password** statement provides the issuing CA with the router certificate's password, as set by the administrator and normally obtained from the SCEP enrollment Web page of the CA. The password is 16 characters in length.

Optionally, the router certificate keypair can be regenerated by using the **re-generate-keypair** statement.

To configure automatic reenrollment properties, include the following statements at the **[edit security pki]** hierarchy level:

```
[edit security pki]
auto-re-enrollment {
    certificate-id {
        ca-profile ca-profile-name;
        challenge-password password;
        re-enroll-trigger-time-percentage percentage;
        re-generate-keypair;
        validity-period days;
    }
}
```

percentage is the percentage for the reenroll trigger time. The range can be from 1 through 99 percent.

days is the number of days for the validity period. The range can be from 1 through 4095.

Tasks to configure automatic reenrollment of certificates are:

Specify the Certificate ID

Use the **certificate-id** statement to specify the name of the router certificate to configure for auto-reenrollment. To specify the certificate ID, include the statement at the **[edit security pki auto-re-enrollment]** hierarchy level:

```
[edit security pki auto-re-enrollment]
certificate-id certificate-name;
```

Specify the CA Profile

Use the ca-profile statement to specify the name of the CA profile from the router certificate previously specified by certificate ID. To specify the CA profile, include the statement at the [edit security pki auto-re-enrollment certificate-id *certificate-name*] hierarchy level:

```
[edit security pki auto-re-enrollment certificate-id certificate-name]
ca-profile ca-profile-name;
```

The referenced ca-profile must have an enrollment URL configured at the [edit security pki ca-profile *ca-profile-name* enrollment url] hierarchy level.

Specify the Challenge Password

The challenge password is used by the CA specified by the *PKI* certificate ID for reenrollment and revocation. To specify the challenge password, include the following statement at the [edit security pki auto-re-enrollment certificate-id *certificate-name*] hierarchy level:

```
[edit security pki auto-re-enrollment certificate-id certificate-name]
challenge-password password;
```

Specify the Reenroll Trigger Time

Use the re-enroll-trigger-time statement to set the percentage of the validity period before expiration at which reenrollment occurs. To specify the reenroll trigger time, include the following statement at the [edit security pki auto-re-enrollment certificate-id *certificate-name*] hierarchy level:

```
[edit security pki auto-re-enrollment certificate-id certificate-name]
re-enroll-trigger-time percentage;
```

percentage is the percentage for the reenroll trigger time. The range can be from 1 through 99 percent.

Specify the Regenerate Keypair

When a regenerate keypair is configured, a new keypair is generated during reenrollment. On successful reenrollment, a new keypair and new certificate replace the old certificate and keypair. To generate a new keypair, include the following statement at the [edit security pki auto-re-enrollment certificate-id *certificate-name*] hierarchy level:

```
[edit security pki auto-re-enrollment certificate-id certificate-name]
re-generate-keypair;
```

Specify the Validity Period

The validity-period statement specifies the router certificate validity period, in number of days, that the specified router certificate remains valid. To specify the validity period, include the statement at the [edit security pki auto-re-enrollment certificate-id *certificate-name*] hierarchy level:

```
[edit security pki auto-re-enrollment certificate-id certificate-name]
  validity-period days;
```

days is the number of days for the validity period. The range can be from 1 through 4095.

RELATED DOCUMENTATION

[Digital Certificates Overview](#)

[Configuring Digital Certificates for an ES PIC](#)

Enroll a CA Certificate Using SCEP

IN THIS SECTION

- [Enroll a CA Certificate Online Using SCEP | 90](#)
- [Example: Enroll a Local Certificate Online Using SCEP | 91](#)
- [Example: Using SCEP to Automatically Renew a Local Certificate | 93](#)

Enroll a CA Certificate Online Using SCEP

Before you begin:

1. Generate a public and private keypair.
2. Create a CA profile.

To enroll a CA certificate online:

1. Retrieve the CA certificate online using SCEP.

```
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 about the certificate, such as the bit length of the keypair, use the command `show security pki ca-certificate`.

Example: Enroll a Local Certificate Online Using SCEP

IN THIS SECTION

- [Requirements | 91](#)
- [Overview | 92](#)
- [Configuration | 92](#)
- [Verification | 93](#)

This example shows how to enroll a local certificate online using SCEP.

Requirements

Before you begin:

- Generate a public and private keypair. See ["Self-Signed Digital Certificates" on page 65](#).
- Configure a CA profile. See ["Example: Configure a CA Profile" on page 62](#).
- For SCEP, enroll the CA certificate. See ["Enroll a CA Certificate Online Using SCEP" on page 90](#).

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 keypair (for example, `qqq`), and the following information:

- The challenge password provided by the CA administrator for certificate enrollment and reenrollment.

The [RFC8894](#) states that the challenge password is optional when an existing certificate signs a renewal request. The challenge password is not mandatory. You can commit the configuration even without challenge password.

- 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 DN 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).

After you obtain the device certificates and the online enrollment begins for the certificate ID, the command is processed asynchronously.

Configuration

IN THIS SECTION

- [Procedure | 93](#)

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 qqq challenge-password ca-provided-password domain-name qqq.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 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 ECDSA key with 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 | 94

- [Overview | 94](#)
- [Configuration | 95](#)
- [Verification | 95](#)

You can use either CMPv2 or 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 31.
- Obtain a local certificate. See "[Example: Enroll a Local Certificate Online Using SCEP](#)" on page 91.

Overview

You can enable the device to automatically renew certificates that you acquired by online enrollment or loaded manually. Automatic certificate renewal saves you from having to remember to renew certificates on the device before certificate expire and helps 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%, then configure 10 for the reenrollment trigger.

For automatic certificate renewal to work, make sure the device is able to reach the CA server and the certificate remain on the device during the renewal process. Furthermore, you must also ensure that the CA that issues 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 keypair. A good security practice is to regenerate a new keypair for reenrollment. To generate a new keypair, use the `re-generate-keypair` command.

Configuration

IN THIS SECTION

- [Procedure | 95](#)

Procedure

Step-by-Step Procedure

To enable and configure local certificate reenrollment:

1. Use the following command 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.

Enroll a CA Certificate Online Using CMPv2

IN THIS SECTION

- [Certificate Enrollment and Reenrollment Messages | 96](#)
- [EE Certificate with Issuer CA Certificate | 96](#)
- [EE Certificate with CA Certificate Chain | 97](#)

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. You must create the CA profile before CA certificates 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.

The CMPv2 server responds with Initialization Response (IP). The response contains the EE certificate along with optional CA certificates. You can verify the message integrity and message authenticity Initialization Response using shared-secret-information according to RFC 4210. You can also verify the Initialization Response using the issuer CA public key. Before you reenroll an EE 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 the trusted CA store. These CA certificates are later used for EE certificate validation.

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

EE Certificate with Issuer CA Certificate

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

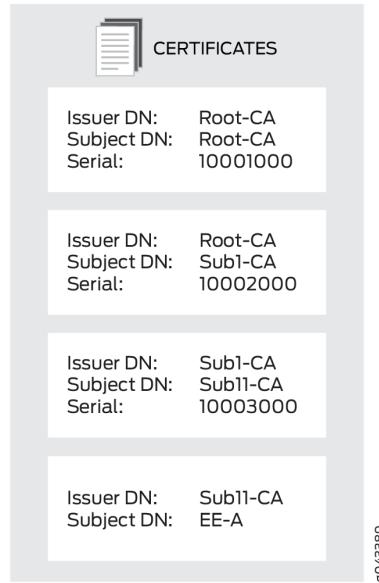
The Initialization Response message can contain an EE certificate as well as a self-signed issuer CA certificate. The EE certificate is first stored in the certificate store, and then the CA certificate is checked. If the CA certificate is found and the subject DN of the CA certificate in the Initialization Response message matches the issuer DN of the EE 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.

EE Certificate with CA Certificate Chain

In many deployments, the EE certificate is issued by an intermediate CA in a certificate chain. In this case, the Initialization Response message can contain the EE 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 EE 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 7 on page 97](#), the Initialization Response message includes the EE certificate and three CA certificates in a certificate chain.

Figure 7: End-Entity Certificate with CA Certificate Chain



In the above image, the EE certificate is stored in the EE 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 EE certificate. It is stored in the CA profile that is specified with the CMPv2 certificate enrollment command.

The device checks the presence of each of the remaining CA certificates in the chain 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 000000010001000.

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.

Install a Digital Certificate on Your Router

IN THIS SECTION

- [Manually Request a Digital Certificate | 98](#)

A digital certificate is an electronic means for verifying your identity through a trusted third party, known as a 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 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 PKI provides an infrastructure for digital certificate management.

Manually Request a Digital Certificate

To obtain digital certificates manually, you must configure a CA profile, generate a private-public keypair, create a local certificate, and load the certificates on the router. After loading the certificates, they can be referenced in your IPsec-VPN configuration.

The following procedure shows how you can configure a CA profile:

1. Use the following command to configure a CA profile:

```
user@R2# set security pki ca-profile entrust ca-identity entrust enrollment url http://
ca-1.example.com/cgi-bin/pkiclient.exe
```

Commit this configuration. The configuration on Router 2 must contain the following:

```
[edit]
      security {
        pki {
          ca-profile entrust {
            ca-identity entrust;
            enrollment {
              url http://ca-1.example.com/cgi-bin/pkiclient.exe;
            }
          }
        }
      }
```

2. The device performs CRL verification by default. You can optionally specify the LDAP server where the CA stores the CRL. The certificate typically includes a certificate distribution point (CDP), which contains information about how to retrieve the CRL for the certificate. The router uses this information to download the CRL automatically. In this example, the LDAP URL is specified, which overrides the location provided in the certificate:

```
user@R2# set security pki ca-profile entrust revocation-check crl url ldap://10.157.90.185/
o=juniper,c=uscertificateRevocationListbase
```

Commit this configuration. The configuration on Router 2 must contain the following:

```
[edit]
      security pki ca-profile entrust {
        revocation-check {
          crl {
            url ldap://10.157.90.185/
          o=juniper,c=uscertificateRevocationListbase;
        }
      }
```

```

        }
    }
```

3. After you configure the CA profile, request a CA certificate from the trusted CA. In this example, the certificate is enrolled online and installed into the router automatically.

```

user@R2> request security pki ca-certificate enroll ca-profile entrust
        Received following certificates:
        Certificate: C=us, O=juniper
        Fingerprint:
        00:8e:6f:58:dd:68:bf:25:0a:e3:f9:17:70:d6:61:f3:53:a7:79:10
        Certificate: C=us, O=juniper, CN=First Officer
        Fingerprint:
        bc:78:87:9b:a7:91:13:20:71:db:ac:b5:56:71:42:ad:1a:b6:46:17
        Certificate: C=us, O=juniper, CN=First Officer
        Fingerprint:
        46:71:15:34:f0:a6:41:76:65:81:33:4f:68:47:c4:df:78:b8:e3:3f
        Do you want to load the above CA certificate ? [yes,no] (no)
yes
```

If you obtain the CA certificate directly from the CA (for example, as an e-mail attachment or website download), you can install it with the `request security pki ca-certificate load` command.

4. Next, you must generate a private-public keypair before you can create a local certificate.

```

user@R2> request security pki generate-key-pair certificate-id local-entrust2
        Generated key pair local-entrust2, key size
        1024 bits
```

When the keypair is available, generate a local certificate request and send it to the CA for processing.

```

user@R2> request security pki generate-certificate-request
        certificate-id local-entrust2 domain-name
        router2.example.com
        filename entrust-req2 subject
        cn=router2.example.com
        Generated certificate request
```

-----BEGIN CERTIFICATE REQUEST-----

```

MIIBoTCCAQoCAQAwGjEYMBYGA1UEAxMPdHAXLmp1bm1wZXIubmV0MIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQCiUFk1Qws1Ud
+AqN5DDxRs2kVyKEhh9qoVFnz+
Hz4c9vsy3B8ElwTJ1kmIt2cB3yifB6zePd
+6WYpf57Crwre7YqPkjXM31F6z3Yjx
H
+1BPNbCxNWYvyrnSyVYDbFj8o0Xyqog8ACDfVL2JBWrPNBYy7imq/K9soDBbAs6

5hZqqwIDAQABoEcwRQYJKoZIhvcNAQkOMTgwNjAOBgNVHQ8BAf8EBAMCB4AwJAYD
VR0RAQH/
BBowGIIWdHAXLmVuZ2xhYi5qdW5pcGVyLm5ldDANBgkqhkiG9w0BAQQF
AA0BgQBc2rq1v5S0QXH7LCb/
FdqAL8ZM6GoaN5d6cGwq4bB6a7UQFgtoH406gQ3G

3iH0Zfz4xMIBpJYuGd1dkqgvcDoH3AgTsLkfn7Wi3x5H2qeQVs9bvL4P5nvEZLND

EIMUHwteolZCiZ70f09Fer9cXWHSQs1UtXtgPqQJy2xIeImLgw==
```

-----END CERTIFICATE REQUEST-----

Fingerprint:

0d:90:b8:d2:56:74:fc:84:59:62:b9:78:71:9c:e4:9c:54:ba:16:97 (sha1)

1b:08:d4:f7:90:f1:c4:39:08:c9:de:76:00:86:62:b8 (md5)

You can request the creation and installation of a local certificate online with the request security pki local-certificate enroll command.

5. The trusted CA digitally signs the local certificate and returns it to you. Copy the certificate file into the router and load the certificate.

```

user@R2> request security pki local-certificate load filename /tmp/router2-cert
certificate-id local-entrust2
                                         Local certificate local-entrust2 loaded
                                         successfully
```

The name of the file sent to you by the CA might not match the name of the certificate identifier. However, the certificate-id name must always match the name of the keypair you generated for the router.

Understand ACME Protocol

SUMMARY

Learn about ACME protocol and how to enroll the certificate.

IN THIS SECTION

- [What is ACME Protocol | 102](#)
- [Enroll Local Certificate Using Let's Encrypt Server | 103](#)
- [Manually Reenroll Local Certificate | 105](#)
- [Delete ACME Account | 105](#)

What is ACME Protocol

IN THIS SECTION

- [Limitations | 103](#)

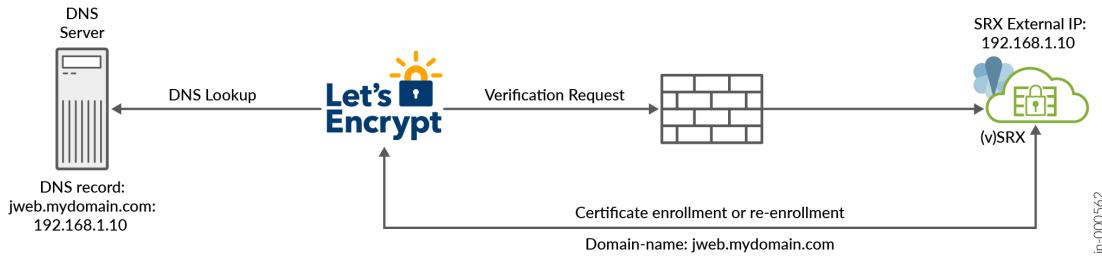
Automated Certificate Management Environment (ACME) protocol is a new PKI enrollment standard used by several PKI servers such as Let's Encrypt. The Let's encrypt certificate allows for free usage of webserver certificates in firewalls, and this can be used in Juniper Secure Connect and J-Web. The Junos OS automatically reenrolls 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 firewalls enroll 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 firewall interface as shown in Figure 2. Let's Encrypt server must be able to reach the firewall interface on TCP port 80. During the certificate enrollment, the firewall temporarily enable this incoming request automatically. If your firewall or an intermediate device or a router is blocking the TCP port 80, certificate enrollment will fail.

The firewall uses port 443 for outbound communication with the [Let's Encrypt Server](#). Ensure that your network security policies allow this outbound traffic. Communication over port 80 to the Let's Encrypt server is not supported for outbound connections from firewall.

Figure 8: Name Resolution for Let's Encrypt



Limitations

- The dns-01 and external account binding are not supported.
- ACME cannot be used when J-Web listens 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. The ACME enrollment process does not include CRL validation.

```
[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
```

To download the **ISRG_Root_X1.pem** certificate, see [KB84991](#).

4. Create ACME key ID.

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

5. Prepare 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.

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 firewall interface as shown in Figure 2. Let's Encrypt server must be able to reach the firewall interface on TCP port 80. During the certificate enrollment, the firewall temporarily enable this incoming request automatically. If your firewall or an intermediate device or a router is blocking the TCP port 80, certificate enrollment will fail.

The firewall uses port 443 for outbound communication with the [Let's Encrypt Server](#). Ensure that your network security policies allow this outbound traffic. Communication over port 80 to the Let's Encrypt server is not supported for outbound connections from firewall.

```
[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 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.

Manually Reenroll Local Certificate

To reenroll a local certificate online:

1. Initiate the reenrollment 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 reenrollment is finished, the issued certificate will be loaded in certificate-id service-mydomain.

Delete ACME Account

To delete the ACME account, perform the following step:

1. Issue the following command.

```
[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.

Platform-Specific SSL Termination Services Behavior

Use [Feature Explorer](#) to confirm platform and release support for specific features.

Use the following table to review platform-specific behaviors for your platform:

Platform	Difference
SRX Series	<ul style="list-style-type: none"> SRX5400, SRX5600, and SRX5800 support the SSL termination services.

Revoke a Certificate

SUMMARY

Learn how to revoke various PKI certificates.

IN THIS SECTION

- Example: Manually Load a CRL onto the Device | [106](#)
- Dynamic CRL Download and Verify | [108](#)
- Example: Configuring a CA Profile with CRL Locations | [111](#)
- Example: Verify Certificate Validity | [113](#)
- Delete a Loaded CRL | [114](#)

Digital certificates have an expiration date; however, before expiration, a certificate might no longer be valid due to many reasons. You can manage certificate revocations and validations locally and by referencing a CA CRL.

Example: Manually Load a CRL onto the Device

IN THIS SECTION

- Requirements | [107](#)
- Overview | [107](#)
- Configuration | [107](#)

- [Verification | 108](#)

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

Requirements

Before you begin:

1. Generate a public-private keypair. See "[Self-Signed Digital Certificates](#)" on page 65.
2. Generate a certificate request. See "[Example: Configure a CA Profile](#)" on page 62.
3. Configure a CA profile. See "[Example: Configure a CA Profile](#)" on page 62.
4. Load your certificate onto the device. See "[Example: Load CA and Local Certificates Manually](#)" on page 77.

Overview

You can load a CRL manually, or you can have the device load it automatically when you verify the 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 you have already loaded a CRL into the `ca-profile`, run `clear security pki crl ca-profile ca-profile-ipsec` the command first to clear the old CRL.

Configuration

IN THIS SECTION

- [Procedure | 108](#)

Procedure

Step-by-Step Procedure

To load a CRL certificate manually:

1. Load a CRL certificate.

```
[edit]
user@host> request security pki crt 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 crt` operational mode command.

Dynamic CRL Download and Verify

Digital certificates are issued for a set period of time. Digital certificates become invalid after the specified expiration date. A CA can revoke an issued certificate by listing it in a 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:

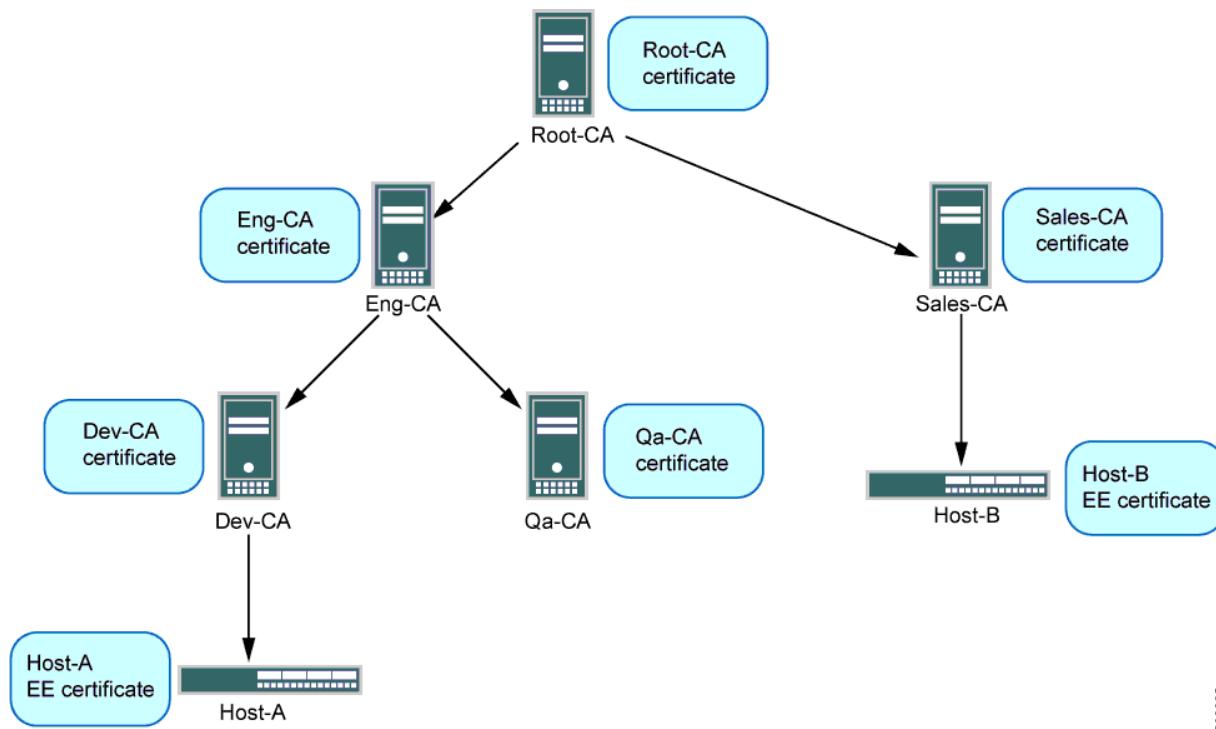
- Allows the local device to download the Dynamic CA and Dynamic CRL (for corresponding CA) as per peer's localcert issuer
- Checks the revocation status of the peer's certificate

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 9 on page 109](#), 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 9: Multilevel Hierarchy for Certificate-Based Authentication



g039005

In case of single hierarchy CA servers or CA certificate chain, the local EE certificate and the received peer EE certificate is 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 11 on page 110](#) provides few sample scenarios where dynamic CA or CRL is not created:

Table 11: 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 profiles 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 9 on page 109](#), 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: Configuring a CA Profile with CRL Locations

IN THIS SECTION

- Requirements | [111](#)
- Overview | [112](#)
- Configuration | [112](#)
- Verification | [112](#)

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

Requirements

Before you begin:

1. Generate a keypair in the device. See "[Digital Certificates](#)" on page 31.
2. Create a CA profile or profiles containing information specific to a CA. See "[Example: Configure a CA Profile](#)" on page 62.
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 75.
4. Load the certificate onto the device. See "[Example: Load CA and Local Certificates Manually](#)" on page 77.
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 106.

Overview

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

Configuration

IN THIS SECTION

- [Procedure | 112](#)

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 | 113](#)
- [Overview | 113](#)
- [Configuration | 113](#)
- [Verification | 114](#)

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 CRL url in the ca-profile configuration. Additionally, you must have network reachability to the same host to receive the checks.

Configuration

IN THIS SECTION

- [Procedure | 114](#)

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

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 CRL.

```
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 a Certificate

SUMMARY

Learn how you can validate the CA certificates..

IN THIS SECTION

- [Validate Digital Certificate on SRX Series Firewall | 115](#)
- [Validate Digital Certificate on MX Series Devices | 121](#)
- [Example: Validating Digital Certificate by Configuring Policy OIDs | 127](#)
- [Example: Improving Digital Certificate Validation by Configuring Policy OIDs on an MX Series Device | 132](#)
- [Example: Configuring a Device for Peer Certificate Chain Validation | 137](#)
- [Configure the Certificate Expiration Trap | 149](#)

Validate Digital Certificate on SRX Series Firewall

IN THIS SECTION

- [Policy Validation | 116](#)
- [Path Length Validation | 118](#)
- [Key Usage | 119](#)
- [Issuer and Subject DN Validation | 120](#)

During IKE negotiation, the PKI 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.

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 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 that you validate, certificate validation fails.

Before policy validation, you must build a certificate chain containing the self-signed root certificate, intermediate CA certificates, and EE certificate. 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 results in successful certificate validation 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. To validate a peer's certificate 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.

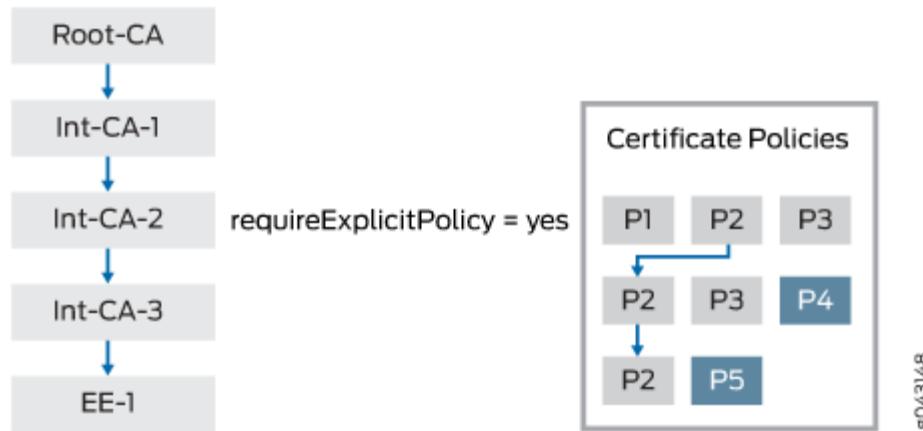
No Policy OIDs Configured on SRX Series Firewalls

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 successful policy validation, a common policy OID must be present in the certificate chain.

[Figure 10 on page 117](#) 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.

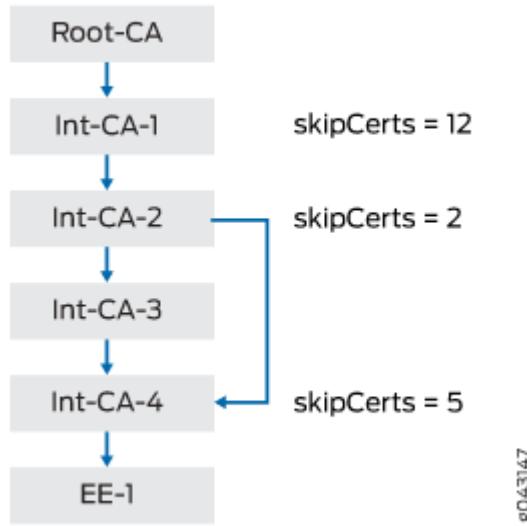
Figure 10: Policy Validation with `requireExplicitPolicy` Field



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 11 on page 118](#) 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 11: 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

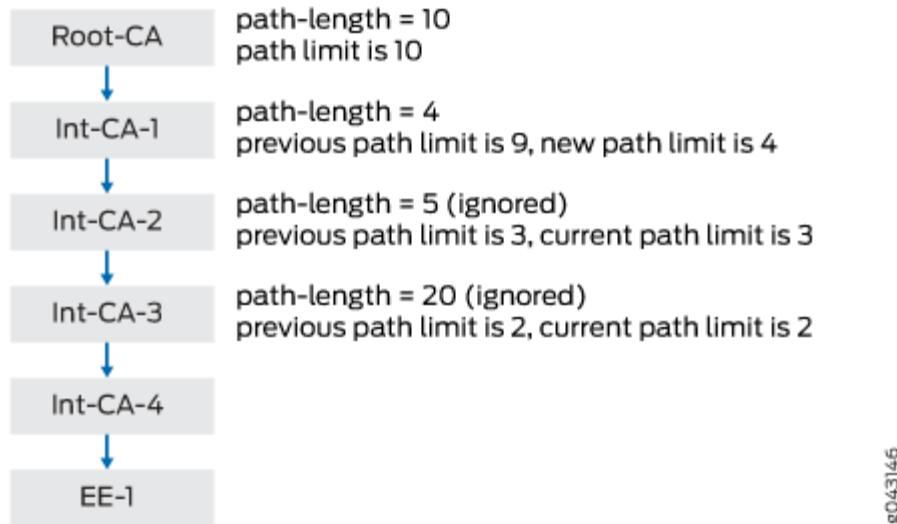
Certificate validation can involve a certificate chain that includes a root CA, one or more optional intermediate CAs, and an EE certificate. The number of intermediate CAs can grow depending upon the deployment scenario. Path length validation provides a mechanism to limit the number of intermediate 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 12 on page 119](#) 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 12: 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, you can sue the key for certificate or CRL signature validation. Because the PKI is responsible for both X509 certificate validation and CRL downloads, you must check the key usage 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 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. You must verify the downloaded CRL files before you download them into the device. One of the verification steps is to validate the CRL signature using a CA certificate. You sign the downloaded CRL with the CA certificate's private key, and you must verify with the private key with the CA certificate's public key stored in the device. The

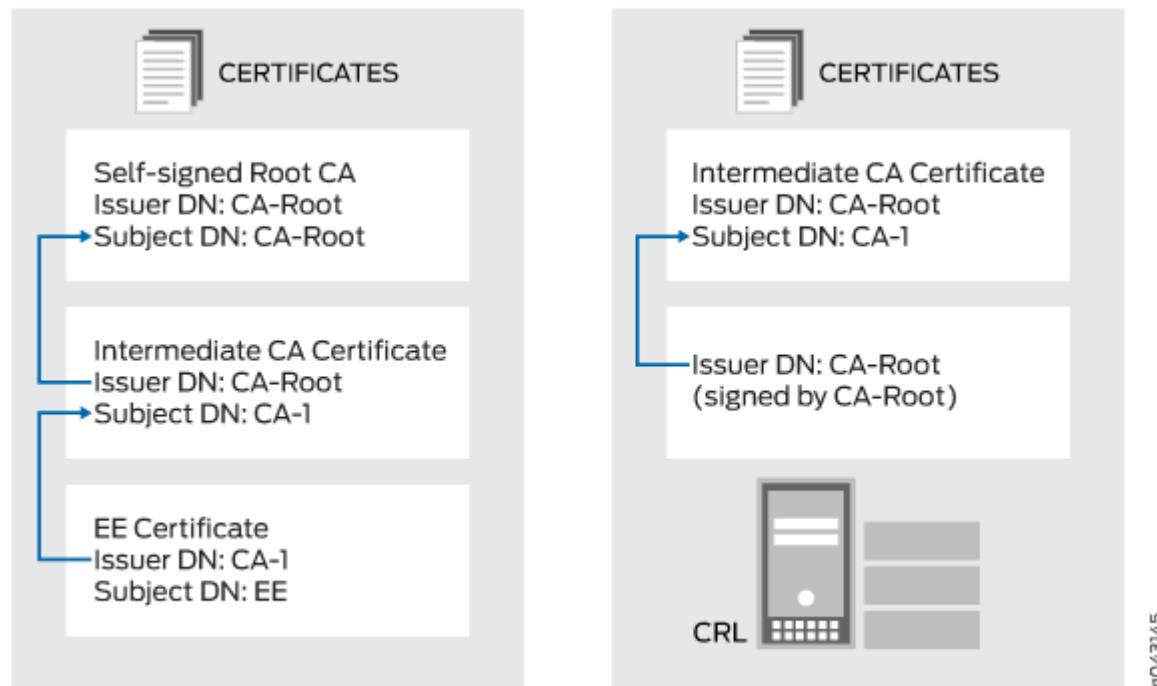
key usage field in the CA certificate must contain the **CRLSign** flag to verify the downloaded CRL. If this flag is not present, the CRL is discarded.

Issuer and Subject DN Validation

You must perform signature validation 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 DN in the certificate or the CRL that is being verified.

[Figure 13 on page 120](#) 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 13: 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 with each other after removing leading and trailing white spaces and converting internal substrings of one or more consecutive whitespaces to a single space.
- Attribute values encoded in types other than `PrintableString` are case-sensitive.

Validate Digital Certificate on MX Series Devices

IN THIS SECTION

- [Policy Validation | 121](#)
- [Path Length Validation | 124](#)
- [Key Usage | 125](#)
- [Issuer and Subject Distinguished Name Validation | 126](#)

Starting in Junos OS Release 16.1R3, MX Series devices support digital certificate validation. During IKE negotiation, the PKI on an MX Series device 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.

Policy Validation

X509 certificates can include optional policy validation fields. If a policy validation field is present, you must perform policy validation for the entire certificate chain including the EE certificate and intermediate 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.

Before performing policy validation, you must build a certificate chain containing the self-signed root certificate, intermediate CA certificate, and EE certificate. 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 MX Series Devices

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 MX Series device.

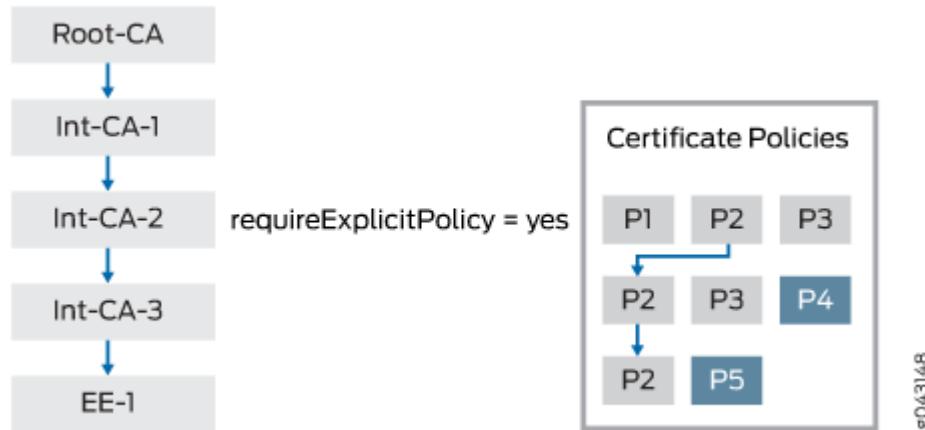
On the MX Series device, you must configure policy OIDs in an IKE policy with the `policy-oids` configuration statement at the `[edit security ike policy policy-name certificate]` hierarchy level. You can configure up to five policy OIDs. To validate a peer's certificate successfully, the peer's certificate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

No Policy OIDs Configured on MX Series Devices

If no policy OID is configured on the MX Series device, policy validation starts whenever the `requireExplicitPolicy` field is encountered in the certificate chain. A certificate might contain one or more certificate policy OIDs. For successful policy validation, a common policy OID must be present in the certificate chain.

[Figure 14 on page 123](#) 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.

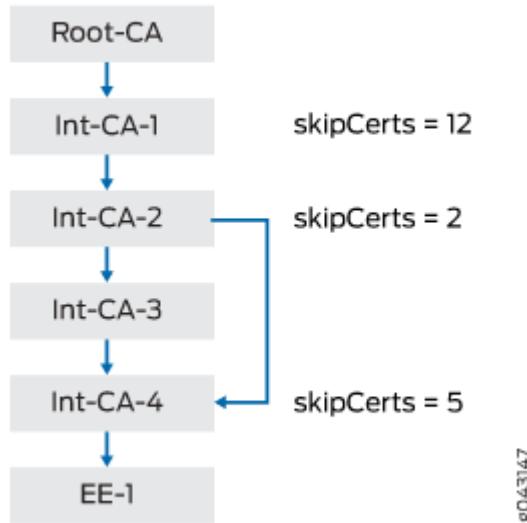
Figure 14: Policy Validation with requireExplicitPolicy Field



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 15 on page 124 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-4 is 5, which is greater than 2, so the Int-CA-4 **skipCerts** value is ignored.

Figure 15: Policy Validation with `skipCerts` Field



8043147

When policy OIDs are configured on the MX Series device, the certificate fields `requireExplicitPolicy` and `skipCerts` are ignored.

Path Length Validation

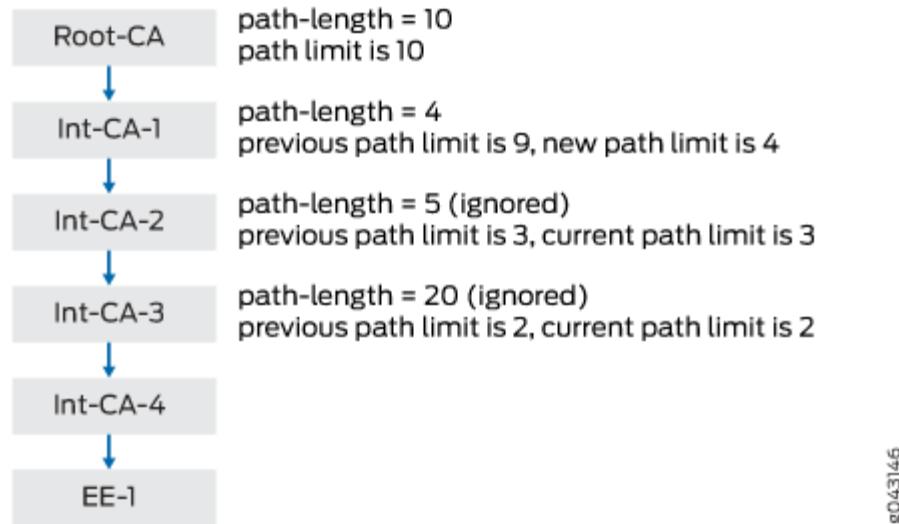
Certificate validation can involve a certificate chain that includes a root CA, one or more optional intermediate CAs, and an EE certificate. The number of intermediate CAs can grow depending upon the deployment scenario. Path length validation provides a mechanism to limit the number of intermediate 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 16 on page 125](#) 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 16: 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.

EE Certificates

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.

CA Certificates

The key can be used for certificate or CRL signature validation. Because the PKI is responsible for both X509 certificate validation and CRL downloads, key usage must be checked before validating the certificate or CRL.

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.

CRL Signature Validation

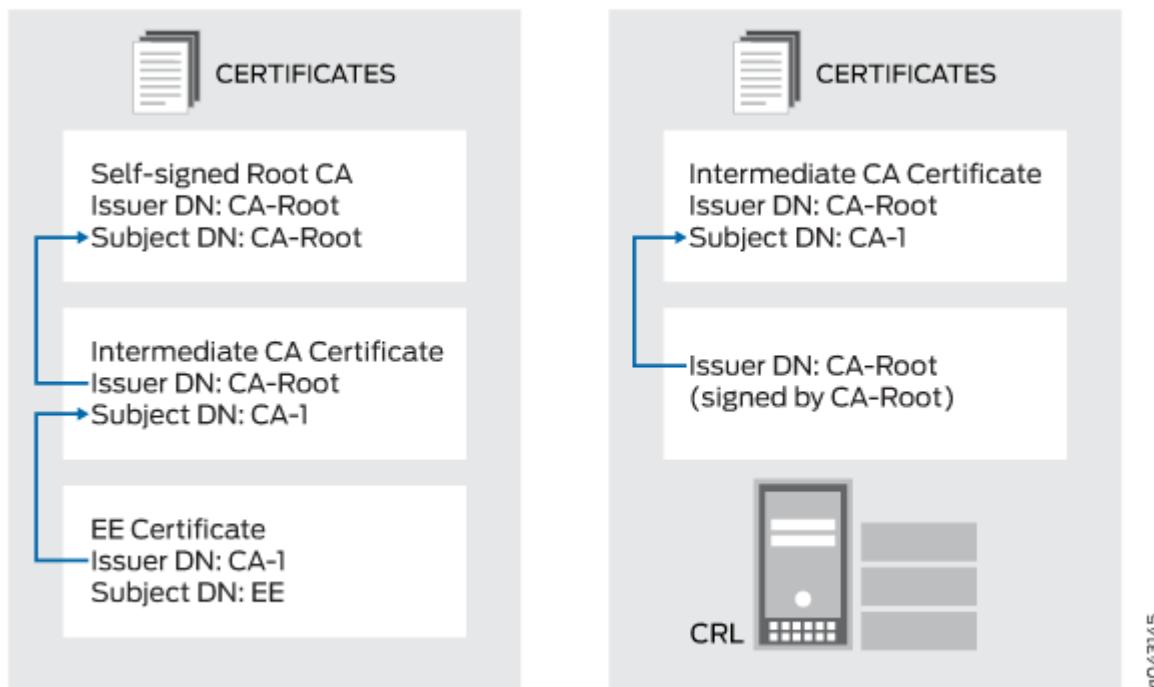
In Phase 1 negotiations, participants check the CRL to see if certificates received during an IKE exchange are still valid. The CRL is periodically downloaded for CA profiles configured with CRL as the certificate revocation check. Downloaded CRL files must be verified before they are downloaded into 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 DN in the certificate or the CRL being verified.

[Figure 17 on page 127](#) 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 17: 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 whites spaces to a single space.
- Attribute values encoded in types other than PrintableString are case-sensitive.

Example: Validating Digital Certificate by Configuring Policy OIDs

IN THIS SECTION

- Requirements | [128](#)
- Overview | [128](#)

- Configuration | [128](#)
- Verification | [130](#)

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 | 129](#)

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 | 130](#)
- [Verifying Policy OID Validation | 131](#)

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` 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 SA are established. If the validation of the peer's certificate fails, no IKE SA 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.

Example: Improving Digital Certificate Validation by Configuring Policy OIDs on an MX Series Device

IN THIS SECTION

- Requirements | [133](#)
- Overview | [133](#)
- Configuration | [133](#)
- Verification | [135](#)

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 MX Series device. This example shows how to configure policy OIDs in the IKE policy on an MX Series device.

You must ensure that at least one of the policy OIDs configured on the MX Series device 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 16.1 or later for MX Series devices.
- Configure an IPsec VPN tunnel.

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 MX Series device is `lc-igloo-root`.

Configuration

IN THIS SECTION

- [Procedure | 133](#)

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 services ipsec-vpn ike policy ike_cert_pol mode main
set services ipsec-vpn ike policy ike_cert_pol proposals ike_cert_prop
```

```
set services ipsec-vpn ike policy ike_cert_pol certificate local-certificate lc-igloo-root
set services ipsec-vpn ike policy ike_cert_pol certificate policy-oids 2.16.840.1.101.3.1.48.2
set services ipsec-vpn 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 *Using the CLI Editor in Configuration Mode* in the [CLI User Guide](#).

To configure policy OIDs for certificate validation:

- Configure the IKE policy:

```
[edit services ipsec-vpn 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 services ipsec-vpn 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 services ipsec-vpn 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 | [135](#)
- Verifying Policy OID Validation | [136](#)

Confirm that the configuration is working properly.

Verifying the CA Certificate

Purpose

Display the CA certificate configured on the device.

Action

From the operational mode, enter the `show security pki ca-certificate ca-profile ca-tmp` 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: 07- 3-2015 10:54 UTC
            Not after: 07- 1-2020 10:54 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 SA are established. If the validation of the peer's certificate fails, no IKE SA is established.

Action

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

```

user@host> show services ipsec-vpn ike security-associations
node0:
-----
Index  State  Initiator cookie  Responder cookie  Mode          Remote Address
821765168 UP    88875c981252c1d8  b744ac9c21bde57e  IKEv2        192.0.2.1
1106977837 UP    1a09e32d1e6f20f1  e008278091060acb  IKEv2        198.51.100.0

```

```

user@host> show services ipsec-vpn 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 500 192.0.2.1

```

```
>213909506 ESP:aes-cbc-192/sha256 8271d2b2 1295/ unlim - root 500 192.0.2.1
<218365954 ESP:aes-cbc-192/sha256 d0153bc0 1726/ unlim - root 1495 198.51.100.0
>218365954 ESP:aes-cbc-192/sha256 97611813 1726/ unlim - root 1495 198.51.100.0
```

Meaning

The `show services ipsec-vpn 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 MX Series device. Check the `policy-oids` values in the peer's certificate chain with the values configured on the MX Series device.

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 MX Series device.

Example: Configuring a Device for Peer Certificate Chain Validation

IN THIS SECTION

- [Requirements | 137](#)
- [Overview | 138](#)
- [Configuration | 139](#)
- [Verification | 146](#)
- [IKE and IPsec SA Failure for a Revoked Certificate | 148](#)

This example shows how to configure a device for certificate chains used to validate peer devices during IKE negotiation.

Requirements

Before you begin, obtain the address of the CA and the information they require (such as the challenge password) when you submit requests for local certificates.

Overview

IN THIS SECTION

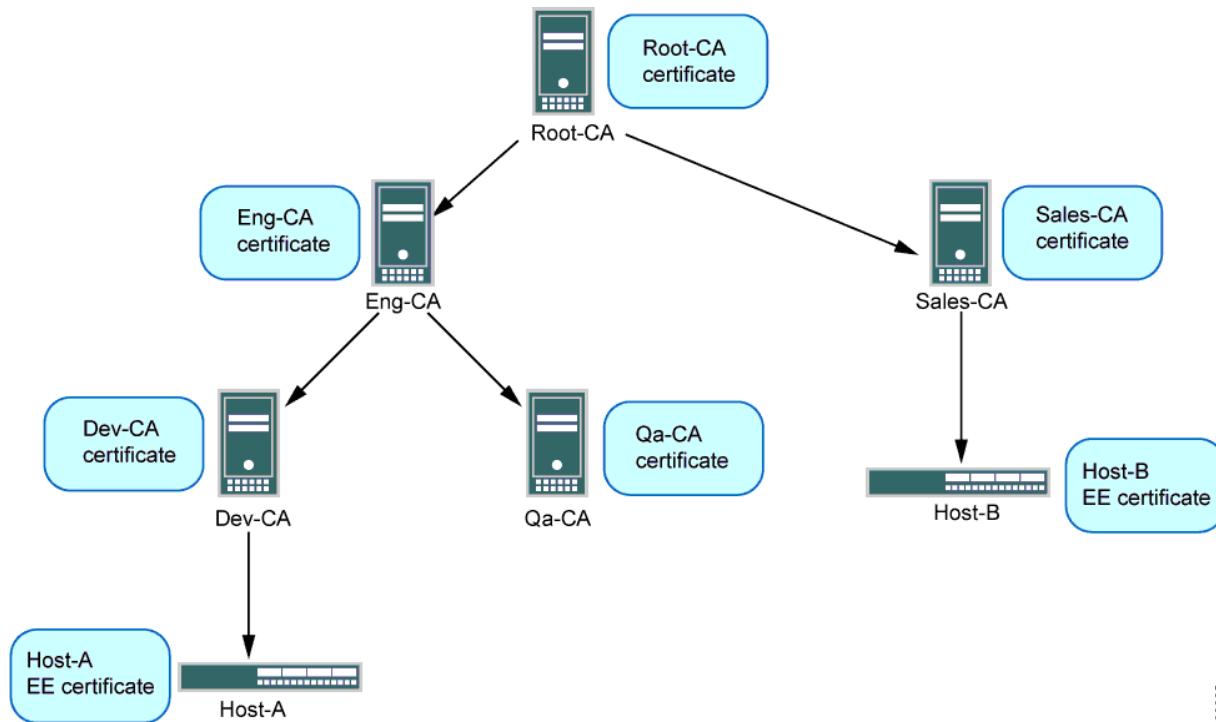
- Topology | 138

This example shows how to configure a local device for certificate chains, enroll CA and local certificates, check the validity of enrolled certificates, and check the revocation status of the peer device.

Topology

This example shows the configuration and operational commands on Host-A, as shown in [Figure 18 on page 138](#). A dynamic CA profile is automatically created on Host-A to allow Host-A to download the CRL from Sales-CA and check the revocation status of Host-B's certificate.

Figure 18: Certificate Chain Example



The IPsec VPN configuration for Phase 1 and Phase 2 negotiation is shown for Host-A in this example. The peer device (Host-B) must be properly configured so that Phase 1 and Phase 2 options are successfully negotiated and SA are established.

Configuration

IN THIS SECTION

- [Configure CA Profiles | 139](#)
- [Enroll Certificates | 141](#)
- [Configure IPsec VPN Options | 144](#)

To configure a device for certificate chains:

Configure CA Profiles

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 pki ca-profile Root-CA ca-identity CA-Root
set security pki ca-profile Root-CA enrollment url http://10.157.88.230:8080/scep/Root/
set security pki ca-profile Root-CA revocation-check use-crl
set security pki ca-profile Eng-CA ca-identity Eng-CA
set security pki ca-profile Eng-CA enrollment url http://10.157.88.230:8080/scep/Eng/
set security pki ca-profile Eng-CA revocation-check use-crl
set security pki ca-profile Dev-CA ca-identity Dev-CA
set security pki ca-profile Dev-CA enrollment url http://10.157.88.230:8080/scep/Dev/
set security pki ca-profile Dev-CA revocation-check use-crl
```

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 *Using the CLI Editor in Configuration Mode* in the [CLI User Guide](#).

To configure CA profiles:

1. Create the CA profile for Root-CA.

```
[edit security pki]
user@host# set ca-profile Root-CA ca-identity CA-Root
user@host# set ca-profile Root-CA enrollment url http://10.157.88.230:8080/scep/Root/
user@host# set ca-profile Root-CA revocation-check use-crl
```

2. Create the CA profile for Eng-CA.

```
[edit security pki]
user@host# set ca-profile Eng-CA ca-identity Eng-CA
user@host# set ca-profile Eng-CA enrollment url http://10.157.88.230:8080/scep/Eng/
user@host# set ca-profile Eng-CA revocation-check use-crl
```

3. Create the CA profile for Dev-CA.

```
[edit security pki]
user@host# set ca-profile Dev-CA ca-identity Dev-CA
user@host# set ca-profile Dev-CA enrollment url http://10.157.88.230:8080/scep/Dev/
user@host# set ca-profile Dev-CA revocation-check use-crl
```

Results

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

```
[edit]
user@host# show security pki
ca-profile Root-CA {
    ca-identity Root-CA;
    enrollment {
        url "http://10.157.88.230:8080/scep/Root/";
    }
    revocation-check {
        use-crl;
    }
}
ca-profile Eng-CA {
```

```
ca-identity Eng-CA;
enrollment {
    url "http://;10.157.88.230:8080/scep/Eng/";
}
revocation-check {
    use-crl;
}
}

ca-profile Dev-CA {
    ca-identity Dev-CA;
    enrollment {
        url "http://;10.157.88.230:8080/scep/Dev/";
    }
    revocation-check {
        use-crl;
    }
}
```

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

Enroll Certificates

Step-by-Step Procedure

To enroll certificates:

1. Enroll the CA certificates.

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

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

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

Type `yes` at the prompts to load the CA certificate.

2. Verify that the CA certificates are enrolled in the device.

```
user@host> show security pki ca-certificate ca-profile Root-CA
Certificate identifier: Root-CA
    Issued to: Root-CA, Issued by: C = us, O = juniper, CN = Root-CA
    Validity:
        Not before: 07- 3-2015 10:54 UTC
        Not after: 07- 1-2020 10:54 UTC
    Public key algorithm: rsaEncryption(2048 bits)
```

```
user@host> show security pki ca-certificate ca-profile Eng-CA
Certificate identifier: Eng-CA
    Issued to: Eng-CA, Issued by: C = us, O = juniper, CN = Root-CA
    Validity:
        Not before: 07- 3-2015 10:54 UTC
        Not after: 07- 1-2020 10:54 UTC
    Public key algorithm: rsaEncryption(2048 bits)
```

```
user@host> show security pki ca-certificate ca-profile Dev-CA
Certificate identifier: Dev-CA
    Issued to: Dev-CA, Issued by: C = us, O = juniper, CN = Eng-CA
    Validity:
        Not before: 07- 3-2015 10:54 UTC
        Not after: 07- 1-2020 10:54 UTC
    Public key algorithm: rsaEncryption(2048 bits)
```

3. Verify the validity of the enrolled CA certificates.

```
user@host> request security pki ca-certificate verify ca-profile Root-CA
CA certificate Root-CA verified successfully
```

```
user@host> request security pki ca-certificate verify ca-profile Eng-CA
CA certificate Eng-CA verified successfully
```

```
user@host> request security pki ca-certificate verify ca-profile Dev-CA
CA certificate Dev-CA verified successfully
```

4. Enroll the local certificate.

```
user@host> request security pki local-certificate enroll certificate-id      Host-A ca-
profile Dev-CA challenge-password juniper domain-name host-a.company.net email host-
a@company.net subject DC=juniper,CN=Host-A,      OU=DEV,O=PKI,L=Sunnyvale,ST=CA,C=US
```

5. Verify that the local certificate is enrolled in the device.

```
user@host> show security pki local-certificate
Issued to: Host-A, Issued by: C = us, O = juniper, CN = Dev-CA
Validity:
    Not before: 07- 3-2015 10:54 UTC
    Not after: 07- 1-2020 10:54 UTC
    Public key algorithm: rsaEncryption(1024 bits)
```

6. Verify the validity of the enrolled local certificate.

```
user@host> request security pki local-certificate verify certificate-id Host-A
Local certificate Host-A verification success
```

7. Check the CRL download for configured CA profiles.

```
user@host> show security pki crl
CA profile: Root-CA
```

```
CRL version: V00000001
CRL issuer: C = us, O = juniper, CN = Root-CA
Effective date: 09- 9-2015 13:08
Next update: 09-21-2015 02:55
```

```
CA profile: Eng-CA
CRL version: V00000001
CRL issuer: C = us, O = juniper, CN = Eng-CA
Effective date: 08-22-2015 17:46
Next update: 10-24-2015 03:33
```

```
CA profile: Dev-CA
CRL version: V00000001
CRL issuer: C = us, O = juniper, CN = Dev-CA
Effective date: 09-14-2015 21:15
Next update: 09-26-2012 11:02
```

Configure IPsec VPN Options

CLI Quick Configuration

To quickly configure IPsec VPN options, 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 services ipsec-vpn ike proposal ike_cert_prop_01 authentication-method rsa-signatures
set services ipsec-vpn ike proposal ike_cert_prop_01 dh-group group5
set services ipsec-vpn ike proposal ike_cert_prop_01 authentication-algorithm sha1
set services ipsec-vpn ike proposal ike_cert_prop_01 encryption-algorithm aes-256-cbc
set services ipsec-vpn ike policy ike_cert_pol_01 mode main
set services ipsec-vpn ike policy ike_cert_pol_01 proposals ike_cert_prop_01
set services ipsec-vpn ike policy ike_cert_pol_01 certificate local-certificate Host-A
set services ipsec-vpn ipsec proposal ipsec_prop_01 protocol esp
set services ipsec-vpn ipsec proposal ipsec_prop_01 authentication-algorithm hmac-sha1-96
set services ipsec-vpn ipsec proposal ipsec_prop_01 encryption-algorithm 3des-cbc
set services ipsec-vpn ipsec proposal ipsec_prop_01 lifetime-seconds 300
set services ipsec-vpn ipsec policy ipsec_pol_01 proposals ipsec_prop_01
set services ipsec-vpn ipsec vpn ipsec_cert_vpn_01 ike ipsec-policy ipsec_pol_01
```

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 *Using the CLI Editor in Configuration Mode* in the [CLI User Guide](#).

To configure IPsec VPN options:

1. Configure Phase 1 options.

```
[edit services ipsec-vpn ike proposal ike_cert_prop_01]
user@host# set authentication-method rsa-signatures
user@host# set dh-group group5
user@host# set authentication-algorithm sha1
user@host# set encryption-algorithm aes-256-cbc
[edit services ipsec-vpn ike policy ike_cert_pol_01]
user@host# set mode main
user@host# set proposals ike_cert_prop_01
user@host# set certificate local-certificate Host-A
```

2. Configure Phase 2 options.

```
[edit services ipsec-vpn ipsec proposal ipsec_prop_01]
user@host# set protocol esp
user@host# set authentication-algorithm hmac-sha1-96
user@host# set encryption-algorithm 3des-cbc
user@host# set lifetime-seconds 300
[edit services ipsec-vpn ipsec policy ipsec_pol_01]
user@host# set proposals ipsec_prop_01
[edit services ipsec-vpn ipsec vpn ipsec_cert_vpn_01]
user@host# set ike ipsec-policy ipsec_pol_01
```

Results

From the configuration mode, confirm your configuration by entering the `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@host# show services ipsec-vpn ike
proposal ike_cert_prop_01 {
```

```
authentication-method rsa-signatures;
dh-group group5;
authentication-algorithm sha1;
encryption-algorithm aes-256-cbc;
}
policy ike_cert_pol_01 {
    mode main;
    proposals ike_cert_prop_01;
    certificate {
        local-certificate Host-A;
    }
}
[edit]
user@host# show services ipsec-vpn ipsec
proposal ipsec_prop_01 {
    protocol esp;
    authentication-algorithm hmac-sha1-96;
    encryption-algorithm 3des-cbc;
    lifetime-seconds 300;
}
policy ipsec_pol_01 {
    proposals ipsec_prop_01;
}
```

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

Verification

IN THIS SECTION

- [Verifying IKE Phase 1 Status | 147](#)
- [Verify IPsec Phase 2 Status | 147](#)

If certificate validation is successful during IKE negotiation between peer devices, both IKE and IPsec SA are established.

Verifying IKE Phase 1 Status

Purpose

Verify the IKE Phase 1 status.

Action

Enter the **show services ipsec-vpn ike security-associations** command from the operational mode.

```
user@host> show services ipsec-vpn ike security-associations

  Remote Address  State          Initiator cookie  Responder cookie  Exchange type
  192.0.2.0       Matured        63b3445edda507fb  2715ee5895ed244d  Main
```

Verify IPsec Phase 2 Status

Purpose

Verify the IPsec Phase 2 status.

Action

Enter the **show services ipsec-vpn ipsec security-associations** command from the operational mode.

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

Service set: ips_ss1, IKE Routing-instance: default

  Rule: vpn_rule_ms_2_2_01, Term: term11, Tunnel index: 1
  Local gateway: 10.0.1.2, Remote gateway: 172.16.0.0
  IPSec inside interface: ms-2/2/0.1, Tunnel MTU: 1500
  UDP encapsulate: Disabled, UDP Destination port: 0

  Direction SPI          AUX-SPI      Mode      Type      Protocol
  inbound   2151932129  0            tunnel    dynamic   ESP
  outbound  4169263669  0            tunnel    dynamic   ESP
```

IKE and IPsec SA Failure for a Revoked Certificate

IN THIS SECTION

- [Checking for Revoked Certificates | 148](#)

Checking for Revoked Certificates

Problem

If certificate validation fails during IKE negotiation between peer devices, check to make sure that the peer's certificate has not been revoked. A dynamic CA profile enables the local device to download the CRL from the peer's CA and check the revocation status of the peer's certificate. To enable dynamic CA profiles, the revocation-check crl option must be configured on a parent CA profile.

Solution

To check the revocation status of a peer's certificate:

1. Identify the dynamic CA profile that will show the CRL for the peer device by entering the **show security pki crl** command from operational mode.

```
user@host> show security pki crl
  CA profile: Root-CA
  CRL version: V00000001
  CRL issuer: C = us, O = juniper, CN = Root-CA
  Effective date: 09- 9-2012 13:08
  Next update: 09-21-2012 02:55

  CA profile: Eng-CA
  CRL version: V00000001
  CRL issuer: C = us, O = juniper, CN = Eng-CA
  Effective date: 08-22-2012 17:46
  Next update: 10-24-2015 03:33

  CA profile: Dev-CA
  CRL version: V00000001
  CRL issuer: C = us, O = juniper, CN = Dev-CA
  Effective date: 09-14-2012 21:15
```

```
Next update: 09-26-2012 11:02
```

```
CA profile: dynamic-001
CRL version: V00000001
CRL issuer: C = us, O = juniper, CN = Sales-CA
Effective date: 09-14-2012 21:15
Next update: 09-26-2012 11:02
```

The CA profile **dynamic-001** is automatically created on Host-A so that Host-A can download the CRL from Host-B's CA (Sales-CA) and check the revocation status of the peer's certificate.

2. Display CRL information for the dynamic CA profile by entering the **show security pki crt ca-profile dynamic-001 detail** command from the operational mode.

Enter

```
user@host> show security pki crt ca-profile dynamic-001 detail
CA profile: dynamic-001
CRL version: V00000001
CRL issuer: C = us, O = juniper, CN = Sub11
Effective date: 09-19-2012 17:29
Next update: 09-20-2012 01:49
Revocation List:
  Serial number          Revocation date
  10647C84                09-19-2012 17:29 UTC
```

Host-B's certificate (serial number 10647084) has been revoked.

Configure the Certificate Expiration Trap

Before you begin:

- Understand how certificates works on VPN. Read [Understanding Certificate Chains](#).

This topic shows how to configure certificate expiration trap and configure the number of days before you generate the trap.

1. Configure the number of days before you generate the trap for all certificates.

```
user@host# set security pki trap all-certificates number-of-days
```

2. Configure the number of days before you generate the trap for a CA certificate.

```
user@host# set security pki trap ca-identity ca-profile-name number-of-days
```

3. Configure the number of days before you generate the trap for a local certificate.

```
user@host# set security pki trap certificate-id certificate-id-name number-of-days
```

4. Confirm your configuration by entering the show security pki trap command.

```
user@host# show security pki trap
certificate-id crt_spk1 {
    30;
}
ca-identity Root-CA {
    30;
}
all-certificates {
    30;
}
```

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
16.1R3	Starting in Junos OS Release 16.1R3, MX Series devices support digital certificate validation.

Update a Certificate

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 | 151](#)
- [Configure Dynamic Update of Trusted CA Certificates | 152](#)

Dynamic Update of Trusted CA Certificates

IN THIS SECTION

- [Processes Involved in Dynamic Update of Trusted CA Bundle | 151](#)

A Junos OS device provides a list of default trusted CA certificates. The Junos OS device manages these certificates dynamically. You can also create a custom list of trusted CA certificates and load CA certificates into the device. But you must manage the custom trusted CA certificates 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 of automatically.
- Addition of new CA to the default trusted CA bundle is immediate without having to wait for the new Junos OS release.

Processes Involved in Dynamic Update of Trusted CA Bundle

Dynamic update of default trusted CA bundle involves the following processes:-

- The Juniper CDN server (<http://signatures.juniper.net/cacert>) hosts the default trusted CA certificates.

- The server hosts a signed copy of the target file and the 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 contains the revision number and date of the default trusted CA bundle.
- Junos OS devices automatically downloads the trusted CA bundles by default. You can either use default or non-default routing instance to connect to the Internet to download and update the default trusted CA certificates.
- The PKI process using the PKID securely downloads the default trusted CA bundle (default-trusted-ca-certs) from the CDN server into the device.

The dynamic update of trusted CA certificates does not make 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 152](#).

- Once you issue the ca-profile-group load command, the PKI process loads the default trusted CA certificates in the background, unblocking the CLI, enabling 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, the PKI process only holds the references to ca-profile names with actual CA certificates already deleted. See ["Configuring Dynamic Updated of Trusted CA Certificates" on page 152](#).
- By default, the PKI process polls the CDN server every 24 hours 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, the 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 152](#).

Configure Dynamic Update of Trusted CA Certificates

IN THIS SECTION

- [Check Connectivity to the CDN Server | 153](#)
- [Enable Automatic Download of Default Trusted CA Certificates | 154](#)

- [Download Default Trusted CA Certificates Automatically | 155](#)
- [Download Default Trusted CA Certificates Manually | 156](#)
- [Check the Download Status of Default Trusted CA Certificates | 157](#)
- [Deactivate Automatic Download of Trusted CA Certificates | 158](#)

Prerequisites

Before you configure the dynamic update of default trusted CA certificates, ensure you meet the following prerequisites:

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

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

Check Connectivity to the CDN Server

IN THIS SECTION

- [Overview | 153](#)
- [Configuration | 154](#)

Overview

Use the following command to check connectivity to the CDN server to download the 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 the operational mode of the Junos OS device, issue the following command.

```
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 | 154](#)
- [Configuration | 154](#)

Overview

Juniper Networks regularly updates the default trusted CA certificates on the Juniper CDN server and you can download the certificates on the Junos OS device. Automatic download of default trusted CA certificates is enabled by default on the 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 a Junos OS device using default configuration settings. See [default-trusted-ca-certs \(Security\)](#) for details about the configuration statement. Downloaded default trusted CA certificates automatically load 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.

Download Default Trusted CA Certificates Automatically

IN THIS SECTION

- [Overview | 155](#)
- [Configuration | 155](#)

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 through the URL `signatures.example.net`.
- Specify the nondefault 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. The 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 Manually

IN THIS SECTION

- [Overview | 156](#)
- [Configuration | 156](#)

Overview

Use the following command to manually download default trusted CA certificates to the Junos OS device from the CDN server. This command is in addition to the automatic download of the default trusted CA certificates 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 the default trusted CA certificates from the operational mode of the Junos OS device, issue the following command.

```
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 | 157](#)
- [Configuration | 157](#)

Overview

Use the following commands to check the download status of default trusted CA certificates on the Junos OS device from the CDN server. These commands display the version number and version date. You can use them 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, issue the following command.

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

2. Use the following command to load default trusted CA certificates:

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

Deactivate Automatic Download of Trusted CA Certificates

IN THIS SECTION

- [Overview | 158](#)
- [Configuration | 158](#)

Overview

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

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

Configuration

Configuration

1. To deactivate automatic download of default trusted CA certificates, use the following command.

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

2. Commit the configuration.

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

Delete a Certificate

IN THIS SECTION

- [Delete a Loaded CRL | 159](#)

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, use the following command.

```
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.

Delete a Loaded CRL

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 CRL.

```
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.

Configuration Statements and Operational Commands

IN THIS SECTION

- [Junos CLI Reference Overview | 160](#)

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Read this guide to learn about the syntax and options that make up the statements and commands. Also 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](#)

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. Copyright © 2026 Juniper Networks, Inc. All rights reserved.